Lecture 2 Examples

UCLA, Econ 10P: Introduction to Python for Economists
Dr. Randall R. Rojas

I. Lists & Dictionaries

1. Given the `list` below, perform the following operations (Hint: use the `.` tab command):

stocks = ["Facebook", "Amazon", "Apple", "Google"]

> a. Reverse the order of the stocks.
> b. Remove Facebook from the list. Note: Did you remove Google instead? Why?
> c. Insert the stock "Twitter" between Amazon and Apple.
> d. Remove Twitter from the list, and check if it was done by using the `in` or `not it` commmand.

In [2]:
```
stocks = ["Facebook", "Amazon", "Apple", "Google"]
```

In [3]:
```
# a
stocks.reverse()
stocks
```

Out[3]:
```
['Google', 'Apple', 'Amazon', 'Facebook']
```

In [6]:
```
# b
#stocks.pop(3)
stocks
```

Out[6]:
```
['Google', 'Apple', 'Amazon']
```

In [8]:
```
# c
stocks.insert(1, "Twitter")
stocks
```

Out[8]:
```
['Google', 'Twitter', 'Apple', 'Amazon']
```

In [19]:
```
# d
stocks.remove("Twitter")
stocks
```

Out[19]:
```
['Facebook', 'Amazon', 'Apple']
```

In [20]:
```
"Twitter" in stocks
```

Out[20]:
```
False
```

2. Given the two lists below, perform the following operations:

$X = [[0, 1, 2], 3]$

$Y = [[4, 5], 6, [7, 8]]$

> a. Output the number 0, 1, 2 from X.
> b. Output the number 1 from X.
> c. Combine the two lists into a new one, call it Z.
> d. Append the values 9 and 10 to Z (compare the `extend` vs. `append` functions).
> e. Output the numbers 6, 7, and 8 from Z.
> f. Remove the values 9 and 10 from Z (compare the `remove` vs. `del` functions).
> g. Output every other element of Z (Hint: use the `[::]` syntax).

In [9]:
```
X = [[0,1, 2], 3]
Y = [[4, 5], 6, [7,8]]
```

In [10]:
```
# a
X[0]
```

Out[10]:
```
[0, 1, 2]
```

In [11]:
```
# b
X[0][1]
```

Out[11]:
```
1
```

In [12]:
```
# c
Z = X + Y
Z
```

Out[12]:
```
[[0, 1, 2], 3, [4, 5], 6, [7, 8]]
```

In [26]:
```
# d
# Note: `extend` is used to append several items to a list
Z.extend([9, 10])
Z
```

Out[26]:
```
[[0, 1, 2], 3, [4, 5], 6, [7, 8], 9, 10]
```

In [27]:
```
# Note: `append` is used to add a single item to the end of the list
# Try running this code a couple of times
Z.append([11,12])
print(Z)
```

```
[[0, 1, 2], 3, [4, 5], 6, [7, 8], 9, 10, [11, 12]]
```

In [13]:
```
# e
Z[3:5]
```

Out[13]:
```
[6, [7, 8]]
```

In [14]:
```
Z
```

Out[14]:
```
[[0, 1, 2], 3, [4, 5], 6, [7, 8]]
```

In [15]:
```
# f
del Z[-2:]
Z
```

Out[15]:
```
[[0, 1, 2], 3, [4, 5]]
```

In [16]:
```
# g --> output every second element
Z[::2]
```

Out[16]:
```
[[0, 1, 2], [4, 5]]
```

In [17]:
```
A=[1,2,34,5,6,6,7,83,4]
```

In [18]:
```
A[::2]
```

Out[18]:
```
[1, 34, 6, 7, 4]
```

3. Given the `tuple` below, perform the following operations:
$U = ([1, 2, 3], 4)$

a. Try assigning the value 100 to the first element of U. Did you get an errror?
b. Assign the value 100 to the contents of the first elemennt of U (Note: Contents can change, elements cannot).
c. What is the ouput of $2 * U$?
d. Compare the `types` of $x = (10)$ vs. $x = (10,)$.
e. What `methods` are available for tuples?

In [29]:
```
U = ([1,2,3],4)
```

In [35]:
```
# a
U[0] = 100
#(If you tried this way, then you got an error)
```

```
---------------------------------------------------------------------------
--
TypeError                                 Traceback (most recent call las
t)
Cell In[35], line 2
      1 # a
----> 2 U[0] = 100

TypeError: 'tuple' object does not support item assignment
```

In [41]:
```
# b
U[0][1] = 2
U
```

Out[41]:
```
([1, 2, 3], 4)
```

In [31]:
```
# c
2*U
```

Out[31]:
```
([100, 2, 3], 4, [100, 2, 3], 4)
```

In [23]:
```
# d
# integer
x = (10)
type(x)
```

Out[23]:
```
int
```

In [25]:
```
# d
# tuple
x = (10,)
type(x)
```

```
(10,)
```

4. Given the `dict` (dictionary) below, perform the following operations:

```
personal_data = {'age': 30, 'pets': [0, 1, 2], 'drinks': ['coffee',
'tea']}
```

a. Output the `keys` and `values` .
b. Check the type of `personal_data` .
c. Output (one by one) the values correpsonding to each key.
d. Change the *age* value to 40.
e. Add a new `key-value` pair, sport = 'Chess' to the dictionary.
f. Remove the `key-value` pair from part (e).

In [18]:
```
personal_data = {'age': 30, 'pets': [0, 1, 2], 'drinks': ['coffee', 'tea']}
```

In [19]:
```
# a keys
personal_data.keys()
```

Out[19]:
```
dict_keys(['age', 'pets', 'drinks'])
```

In [20]:
```
# a values
personal_data.values()
```

Out[20]:
```
dict_values([30, [0, 1, 2], ['coffee', 'tea']])
```

In [21]:
```
# b
type(personal_data)
```

Out[21]:
```
dict
```

In [22]:
```
# c
personal_data['age']
```

Out[22]:
```
30
```

In [23]:
```
personal_data['pets']
```

Out[23]:
```
[0, 1, 2]
```

In [24]:
```
personal_data['drinks']
```

Out[24]:
```
['coffee', 'tea']
```

In [25]:
```
# d
personal_data['age'] = 40
```

In [26]:
```
# e
personal_data['sport'] = 'Chess'
```

In [27]:
```
personal_data
```

Out[27]:
```
{'age': 40, 'pets': [0, 1, 2], 'drinks': ['coffee', 'tea'], 'sport': 'Che
ss'}
```

In [28]:
```
# f
del personal_data['sport']
```

In [29]:
```
personal_data
```

Out[29]:
```
{'age': 40, 'pets': [0, 1, 2], 'drinks': ['coffee', 'tea']}
```

II. Functions

1. Write a function that returns the maximum number between two numbers, squares it, and then takes mod 2.

In [3]:
```
def num_fun(x,y):
    return (max(x,y)**2)%2
```

In [4]:
```
num_fun(2,7)
```

Out[4]:
```
1
```

2. Write a function that outputs the volume and surface area of a cube when the length of the side is provided.

In [26]:
```
def cube_vol_surf(length):
    vol = length**3
    surf = 6*(length**2)
    return vol, surf
```

In [27]:
```
cube_vol_surf(10)
```

Out[27]:
```
(1000, 600)
```

3. Write a function that multiplies a list of numbers.

In [28]:
```
def multiply(numbers):
 tot = 1
 for i in numbers:
     tot *= i
 return tot
```

In [39]:
```
multiply((1,2,3))
```

Out[39]:
```
6
```

4. Wtite a function that counts the number of upper case and lower case letters in a string. Hint: Use a `dict` and the `.isupper` function.

In [29]:
```
def string_case(string):
# Note: This example/solution is an adaptation from Analytics Vidhya
    d={"Upper":0, "Lower":0}
    for character in string:
         if character.isupper():
              d["Upper"]+=1
         elif character.islower():
              d["Lower"]+=1
         else:
              pass
    print ("String Input: ", string)
    print ("Number of Upper case characters: ", d["Upper"])
    print ("Number of Lower case Characters: ", d["Lower"])
```

In [30]:
```
string_case("I am Going To Test My Code")
```

```
String Input:  I am Going To Test My Code
Number of Upper case characters:  6
Number of Lower case Characters:  14
```

5. Write a function called `lottery` that solves the following problem (Berk & DeMarzo, Problem 4.8 -same one from the lecture). You are the lucky winner of the $\$30$ million state lottery. You can take your prize money either as (a) 30 payments of $\$1$ million per year (starting today), or (b) $\$15$ million paid today. If the interest rate is 8%, which option should you take?

In [31]:
```
def lottery(C,r, n):
    PV = (C/r)* (1 - (1/(1+r)**n))
    annuity = PV + 1e6
    cash = 15e6
    print(" The annuity is =  $",annuity)
    print(" The cash value is = $15M")
    if annuity>cash:
        print("The annuity is the best option")
    else:
        print("The $15M cash option is better")
```

In [32]:
```
lottery(1000000,.08,29)
```

```
 The annuity is =  $ 12158406.010577684
 The cash value is = $15M
The $15M cash option is better
```

6. Create an example for the use of (i) `try-except` and (ii) `pass`

In [9]:
```
# (i) try-except
text = ('A','1','3.14','55.a')
for t in text:
    try:
        temp = float(t)
        print(temp)
    except ValueError:
        print('Not convertable to a float')
```

```
Not convertable to a float
1.0
3.14
Not convertable to a float
```

In [10]:
```
# (ii) pass
x=9
if x<0:
    print('negative')
elif x==0:
    # you can have the code do something here
    pass
else:
    print('positive')
```

```
positive
```