Python

Python

- השפה אינה צורכת קומפילציה
- בד"כ נדרשות פחות שורות קוד
- אין הצהרה על משתנים
- שפה דינמית
- מאפשרת ניתוח נח של הנתונים
- ה- SCOPE נקבע ע"י הזחה (דוגמאות בהמשך)

Python

- ניתן להוריד python בעזרת הקישור הבא - https://www.python.org/downloads/

- לחילופין, ניתן להשתמש באונליין טרמינל - http://rextester.com/l/python3_online_compiler (יש עוד..)

Python

לשפה יש שני מצבים בסיסיים:

1. **Interactive Mode Programming**

מריצים את הקוד ב- command line אשר מאפשר משוב מיידי תוך שימוש בזכרון פעיל

2. **Script Mode Programming**

שומרים את הקוד בקובץ עם סיומת py ואז מריצים ב- python Interperter

Python

- שמות משתנים ואובייקטים מתחילים באותיות A-Z a-z או _
- אח"כ יכולים להופיע גם מספרים
- לא ניתן להשתמש בתווים %, $, @
- השפה היא case sensitive
- בד"כ כל שורה היא פקודה, אם רוצים לפרוש את הפקודה על יותר משורה אחת יש להוסיף את התו \
- לפקודות המכילות סוגריים (()[],{}) לא צריכים להוסיף את תו מעבר השורה
- גרשיים- מתקבל גרש יחיד, כפולים וגם משולשים
- הערות- עם התו #

Datatypes

Numbers »

String »

List »

Tuple »

Dictionary »

... »

הצבת ערכים למשתנים

- אין צורך בהצהרה לפני ההצבה
- הצבה עם הסימן =
- הצבה מרובה

a=b=c=1

a, b, c = 1, 2.0, ”john”

- מספרים –
  - Int >
  - long >
  - Float >

String

str = 'Hello World!’

- print (str)

  Hello World!
- print (str[0])

  H
- print (str[2:5])

  llo
- print (str[2:])

  llo World!
- print (str * 2)

  Hello World!Hello World!
- print (str + "TEST“)

  Hello World!TEST

Lists

list = ['abcd', 786 , 2.23, 'john', 70.2]

tinylist = [123, 'john’]

- print (list)

  ['abcd', 786, 2.23, 'john', 70.2]
- print (list[0])

  abcd
- print (list[1:3])

  [786, 2.23]
- print (tinylist * 2)

  [123, 'john', 123, 'john’]
- print (list[2:])

  [2.23, 'john', 70.2]
- print (list + tinylist)

  ['abcd', 786, 2.23, 'john', 70.2, 123, 'john’]

Tuples

- דומה למערך אך ללא האפשרות לשינוי או הוספה.
- משמש כ-read only lists

```
tuple = ('abcd', 786 , 2.23, 'john', 70.2)
tinytuple = (123, 'john')
```

- print (tuple)
  ('abcd', 786, 2.23, 'john', 70.2)
- print (tuple[0])
  abcd
- print (tuple[1:3])
  (786, 2.23)
- print (tuple[2:])
  (2.23, 'john', 70.2)
- print (tinytuple * 2)
  (123, 'john', 123, 'john')
- print (tuple + tinytuple)
  ('abcd', 786, 2.23, 'john', 70.2, 123, 'john')

מערכים מקוננים

arr = (4, [5,4, [5.2, 9.3], "What?"], "Yes", ("It", 3, "Ok"))

» print (arr[1][2][1])

9.3

Dictionary

```
dict = {}
dict['one'] = "This is one"
dict[2]     = "This is two"

tinydict = {'name': 'john','code':6734, 'dept': 'sales'}
```

» print (dict['one’])
  This is one
» print (dict[2])
  This is two
» print (tinydict)
  {'name': 'john', 'code': 6734, 'dept': 'sales'}
» print (tinydict.keys())
  {dict_keys(['name', 'code', 'dept'])
» print (tinydict.values())
  dict_values(['john', 6734, 'sales'])

אופרטורים

- השוואה-
  - ==, !=, >,<,>=,<=
- השמה-
  - .... c*= ,a+=
- לוגיים-
  - And, or, not
- Membership and identity
  - in, not in, is, is not

אופרטורים - מתמטיים

- + הוספה
- - הפחתה
- * הכפלה
- / חילוק
- % שארית
- ** חזקה
- // חילוק ועיגול למס' השלם הנמוך

אופרטורים - בינריים

- and - &
- or - |
- xor - ^

Conditional Statement

```
var = 100
if var < 200:
   print ("Expression value is less than 200")
   if var == 150:
      print ("Which is 150")
   elif var == 100:
      print ("Which is 100")
   elif var == 50:
      print ("Which is 50")
elif var < 50:
   print ("Expression value is less than 50")
else:
   print ("Could not find true expression")

print ("Good bye!")
```

Results:

Expression value is less than 200
Which is 100
Good bye!

Conditional Statement

» Short way -

```
b = 10 if a < 5 else 3
#equivalent to b=(a<5?10:3)
```

Loops - While

```
count = 0
while (count < 9):
   print ('The count is:', count)
   count = count + 1
else:
   print (count, " is not less than 9")
print ("Good bye!")
```

Results:

The count is: 0
The count is: 1
The count is: 2
The count is: 3
The count is: 4
The count is: 5
The count is: 6
The count is: 7
The count is: 8
9 is not less than 9
Good bye!

Loops - For

```
fruits = ['banana', 'apple', 'mango’, ‘melon’, ‘watermelon’]
for fruit in fruits:
  print ('Current fruit :', fruit)
  if fruit==‘melon’:
    break;
```

אותו הדבר בדרך אחרת:

```
for index in range(len(fruits)):
  print ('Current fruit :', fruits[index])
  if fruits[index] ==‘melon’:
    break;
```

Results:

Current fruit : banana
Current fruit : apple
Current fruit : mango
Current fruit : melon

Loops - For

דוגמא נוספת:

```
a = [x*x for x in range (1,10)]
print(a)
```

[1, 4, 9, 16, 25, 36, 49, 64, 81]

Functions

- פונקציה מתחילה עם def ואז שם הפונקציה, סוגריים ובסוף נקודותיים
- בתוך הסוגריים מגדירים את משתני הקלט
- בלוק הפונקציה חייב להיות מוזח
- השורה הראשונה בבלוק הפונקציה יכולה להכיל מחרוזת המתארת את הפונקציה
- הפקודה return- גורמת ליציאה מהפונקציה ויכולה גם להחזיר ערך

Functions

```
def printme( str ):
  "This prints a passed string into this function"
  print (str)
  return;
```

» printme("Bye Bye!!")

Bye Bye!!

```
a = [5, 2.3, printme, "Bye"]
```

» print(a[2])

<function printme at 0x00000145930106A8>

» a[2]("what?")

what?

Anonymous Functions

```
sum = lambda arg1, arg2: arg1 + arg2;
» print ("Value of total : ", sum( 10, 20 ))
        Value of total :  30
f = [7, lambda x : x*x , "Hi"]
» print(f[1])
        <function <lambda> at 0x0000017B39DD0D08>
» print(f[1](3))
        9
def o(f,g):
   return lambda x: f(g(x))
b = o(lambda x: x*x, lambda x: x+1)
» print (b(3))
        16
```

תרגיל -

» הדפיסו את כל המספרים הזוגיים במערך

```
» arr =[1,2,3,4,5,6,7,8,9]
for i in arr:
        if i%2 == 0:
                print(i)
```

תרגיל -

- ממשו פונקציית מיון בועות המקבלת מערך (המכיל ביטוי למבדא בתור ערך ראשון) וממיינת את שאר המערך לפי הקומפרטור שנמצא בתחילתו.

פתרון -

```
» arr =[lambda x,y: x-y, 1,6,2,4,5,3,8,9,7]
for i in range(1,len(arr)):
        for j in range(1,len(arr)-i):
                if arr[0](arr[j], arr[j+1]) > 0:
                        temp = arr[j]
                        arr[j] = arr[j+1]
                        arr[j+1] = temp
print (arr)
```

תרגיל -

- כתוב פונקציה המקבלת מספר ומחזירה true אם המספר ראשוני. אחרת מחזירה false.

```
import math
def prime(x):
    for i in range(2,int(math.sqrt(x))+1):
        if x%i == 0:
            return False
    return True
print (prime(49))
```

תרגיל -

- כתוב תכנית הקולטת הודעה מהמשתמש ובודקת האם ההודעה מכילה את הרצף 'EOF'.

```
def isContainEOF(a):
    for i in range(len(a)):
        if a[i:i+3] == 'EOF':
            return True
return False

a = input()
print (isContainEOF(a))
```

תרגיל -

- כתוב תכנית הקולטת מהמשתמש מספר שלם המייצג אורך צלע של מלבן. על התכנית להדפיס שני מלבנים (עשויים מ*) בעלי צלע אחת באורך המספר שנקלט וצלע שנייה באורך 2 יותר מהמספר שנקלט.

פתרון -

```
size = int(input('insert a number:'))

print('*'* (size))
for i in range(0,size):
    print ('*' + ' '*(size-2)+'*')
print('*'* (size))
print('\n')

print('*'* (size+2))
for i in range(0,size-2):
    print ('*' + ' '*(size)+'*')
print('*'* (size+2))
```

Spark

Spark

- Spark מאפשר לבצע חישובים תוך שמירה על רמה גבוהה של מקביליות, וניצול מרבי של משאבי המחשב. הביצועים של Spark באים לידי ביטוי במיוחד על מחשבים מרובי ליבות.

- חישובים אופייניים שנבצע באמצעות Spark הם חישובים המורכבים מהרבה צעדים בלתי תלויים ביניהם שירוצו במקביל ולבסוף נאחד את התוצאות.

Resilient Distributed Dataset (RDD)

- הגדרה- אוסף נתונים גמיש ומבוזר.
- אוסף יחידות או פריטים מקיימים ביניהם חוסר תלות ולכן ניתנים לחישוב מקבילי.
- לדוגמא: שורות שונות בקובץ, אוסף של נתוני סטודנטים וכו'
- אפשר ליצור RDD מקובץ או מרשימה קיימת באמצעות שימוש בפונקציות המיועדות לכך ב Spark
- ניתן לבצע שרשרת של טרנספורמציות כאשר כל אחת מהן מניבה RDD חדש. ובסוף נבצע איסוף לתוך מבנה נתונים פשוט כמו רשימה. (ע"י collect())

Word Count

» The file story.txt was obtained from https://s3.amazonaws.com/text-datasets/nietzsche.txt):

```
>>> text_file = sc.textFile("myDir/sto
>>> word_counts = text_file.flatMap
line.split(" ")) \
                .map(lambda word: (wo
                .reduceByKey(lambda a
                .collect()
>>> for word,count in word_counts:
        print("the word: \"%s\" appe
%(word,count))
```

Result:

```
…
the word: "retrograde" appears 1 time(s)
the word: "grounds" appears 4 time(s)
the word: "VOUS" appears 2 time(s)
the word: ""Flatterers" appears 1 time(s)
the word: "injustice;" appears 1 time(s)
the word: "reciprocity," appears 1 time(s)
the word: "inflicted" appears 2 time(s)
the word: "limbs." appears 1 time(s)
the word: "christened" appears 2 time(s)
the word: "majority--where" appears 1 time(s)
the word: "three-fourths" appears 1 time(s)
the word: "dish," appears 1 time(s)
the word: "73." appears 1 time(s)
the word: "ENVIRONMENT," appears 1 time(s)
the word: ""honesty";" appears 1 time(s)
…
```

Sort by Key

To sort by the key we simply add a call to »
SortByKey (before we call colle

```
word_counts = text_file.flatMap(lan
line.split(" ")) \
            .map(lambda word: (w
            .reduceByKey(lambda a
            .sortByKey() \
            .collect()
for word,count in word_counts[0:15
            print("the word: \"%s\" appears %d time(s)"
%(word,count))
```

Result:

```
the word: "" appears 2032 time(s)
the word: ""=Man" appears 1 time(s)
the word: ""A" appears 2 time(s)
the word: ""AWAY" appears 1 time(s)
the word: ""Ah," appears 1 time(s)
the word: ""All" appears 1 time(s)
the word: ""And" appears 2 time(s)
the word: ""Another" appears 1 time(s)
the word: ""Are" appears 2 time(s)
the word: ""BIG" appears 1 time(s)
the word: ""BY" appears 1 time(s)
the word: ""Bad!" appears 1 time(s)
the word: ""Be" appears 1 time(s)
the word: ""Better" appears 1 time(s)
the word: ""Beyond" appears 1 time(s)
```

Sort by Value

» To sort by value we swap the key and the value and then sort by the key and then swap them back. We sort in **descending** order:

```
> word_counts = (
        text_file.flatMap(lambda line
            .map(lambda word: (word,
            .reduceByKey(lambda a,b:
            .map(lambda (x,y): (y,x))
            #False is for descending or
            .sortByKey(False)
            .map(lambda (x,y): (y,x))
            .collect()
            )
> for word,count in word_counts[0:15]:
        print("the word: \"%s\" appears %d time(s)"
%(word,count))
```

Result:

```
the word: "the" appears 5839 time(s)
the word: "of" appears 4560 time(s)
the word: "and" appears 3562 time(s)
the word: "to" appears 2716 time(s)
the word: "" appears 2032 time(s)
the word: "in" appears 1995 time(s)
the word: "a" appears 1896 time(s)
the word: "is" appears 1857 time(s)
the word: "that" appears 1242 time(s)
the word: "as" appears 1172 time(s)
the word: "it" appears 908 time(s)
the word: "for" appears 808 time(s)
the word: "which" appears 783 time(s)
the word: "be" appears 740 time(s)
the word: "with" appears 665 time(s)
```

Bi-grams

« התייחסות ל-2 מילים צמודות בטקסט

"I read a book about the history of America"

"I read"

"read a"

"a book"

"book about"

"about the"

"the history"

"history of"

"of America"

Tri-grams

« התייחסות ל-3 מילים צמודות בטקסט

"I read a book about the history of America"

"I read a"

“read a book"

“a book about

“book about the”

“about the history”

“the history of"

“history of America”

מימוש Bi-grams ב-Spark

- נשתמש בפונקציה מובנת **zip()**
- הפונקציה מקבלת 2 או יותר רשימות ומחזירה רשימה של זוגות נתונים משתי הרשימות

```
a=[1,2,3]
b=['a','b','c']

zipped = zip(a,b)
» print(list(zipped))
```

[(1,'a'),(2,'b'),(3,'c')]

מימוש Bi-grams ב-Spark

```
def BiGram(line):
    words = line.split()
    return zip(words, words[1:])

pairs = text_file.flatMap(BiGram)
count = pairs.map
                .red
print(count.collec
```

Result:

[((u'against', u'itself--still'), 1), ((u'form', u'or'), 2), ((u'Duhring', u'and'), 1), ((u'every', u'kind'), 2), ((u'This', u'crushing'), 1), ((u'Indeed,', u'if'), 1), ((u'species', u'who'), 1), ((u'also', u'emphasized,'), 1), ((u'lacking', u'in'), 9), ((u'solitude?--the', u'skepticism'), 1)]

מימוש Bi-grams ב-Spark

```
print(count.map(lambda (x,y): (y,x)) \
            .sortByKey(False) \
            .map(lambda (x,y): (y,x)) \
            .collect()[0:10])
```

Result:

[((u'of', u'the'), 910), ((u'in', u'the'), 498), ((u'to', u'the'), 327), ((u'it', u'is'), 240), ((u'to', u'be'), 186), ((u'of', u'a'), 171), ((u'and', u'the'), 157), ((u'for', u'the'), 149), ((u'that', u'the'), 138), ((u'is', u'the'), 131)]

תרגיל -

- ממשו את פונקצית N_Grams המקבלת רשימה ומספר n

```
» def func (a,n):
        return zip(*[a[i:] for i in range(n)])
print(list(func([1,2,3,4,5,6,7,8,9], 4)))
```