HAB 619 Introduction to Scientific Computing in Sports Science

De Motu Animalium G.Borelli (1680)

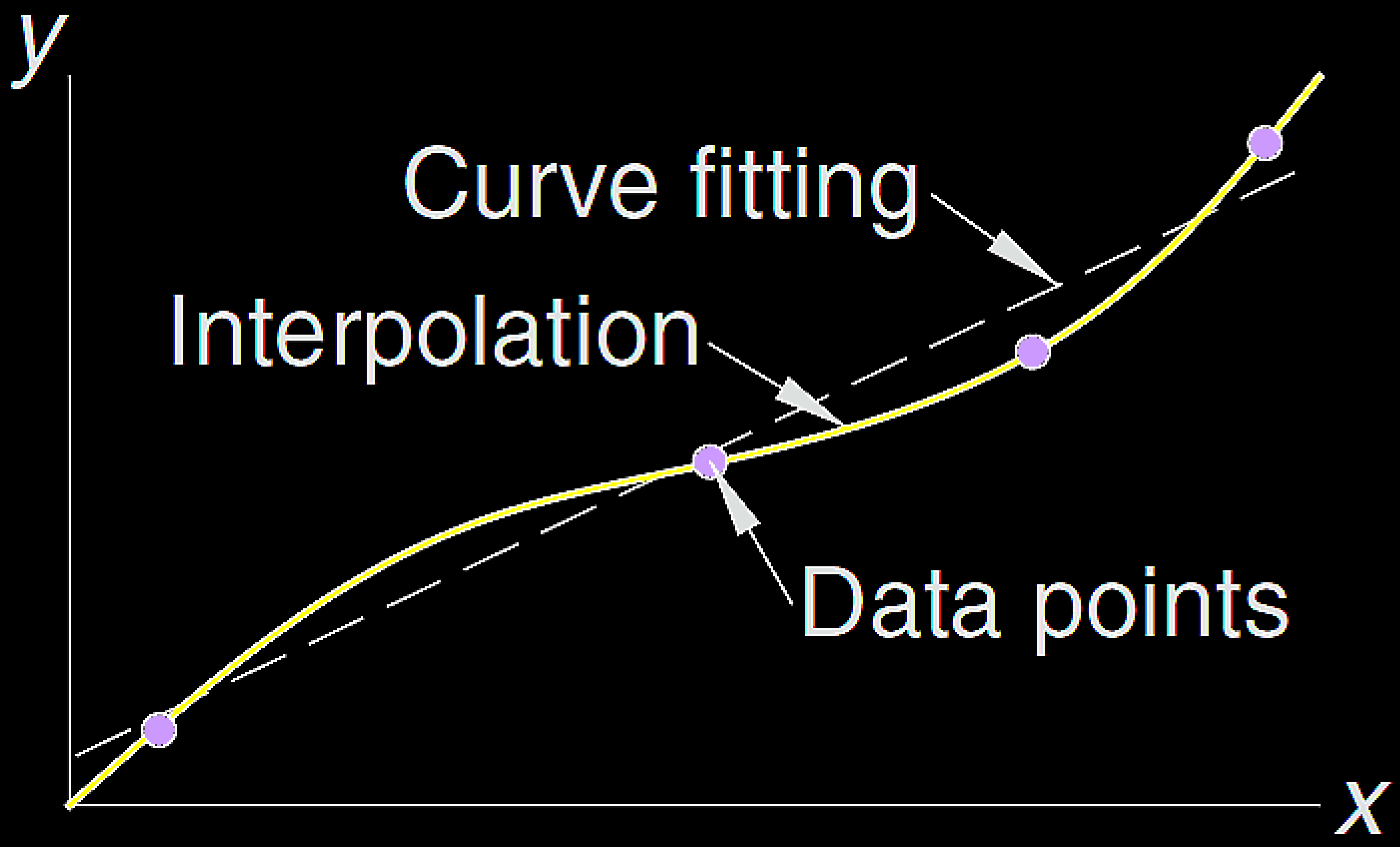
#9

SERDAR ARITAN

serdar.aritan@hacettepe.edu.tr

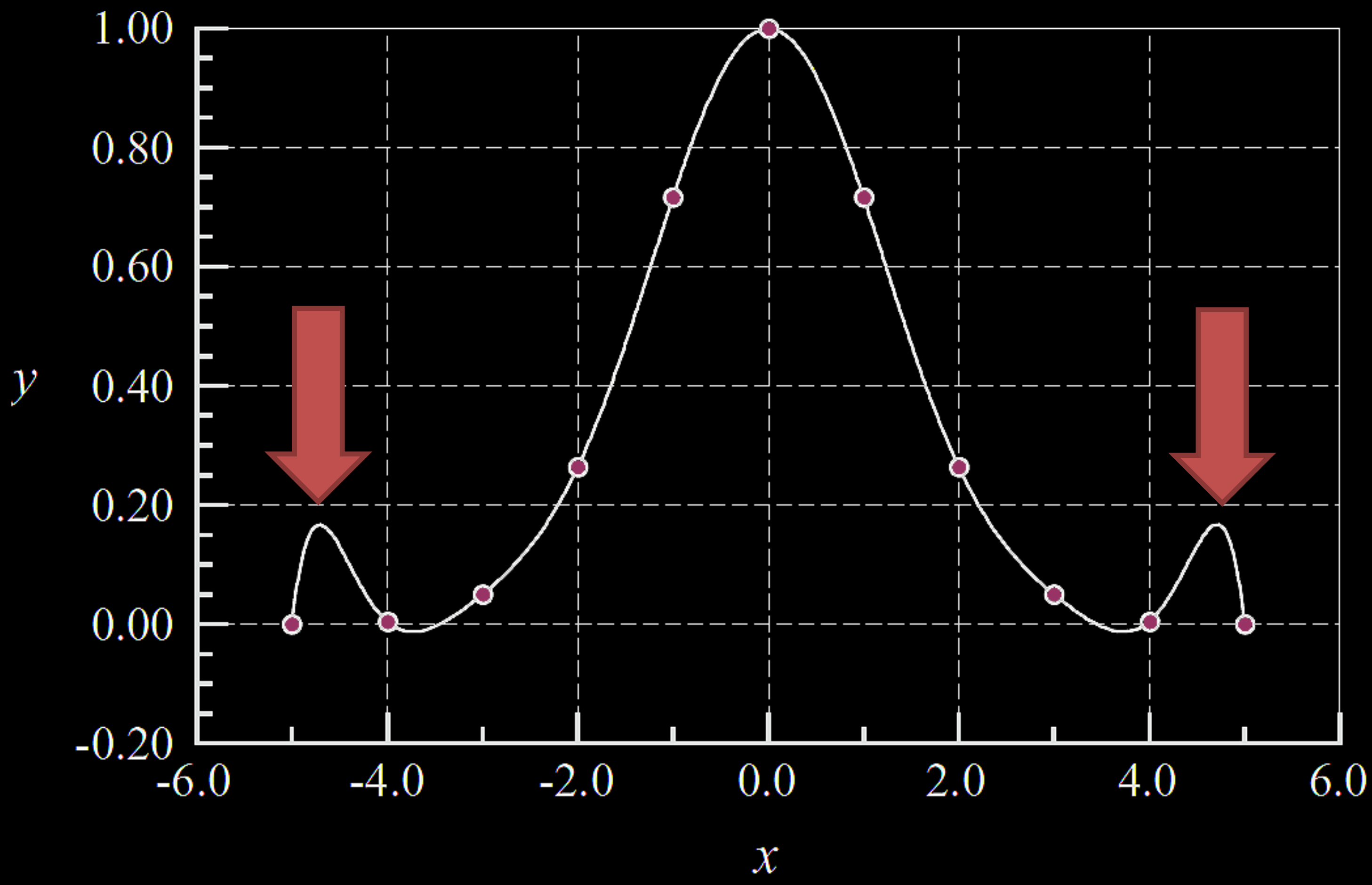
Biyomekanik Araştırma Grubu
www.biomech.hacettepe.edu.tr
Spor Bilimleri Fakültesi
www.sbt.hacettepe.edu.tr
Hacettepe Universitesi, Ankara, Türkiye
www.hacettepe.edu.tr

Interpolation



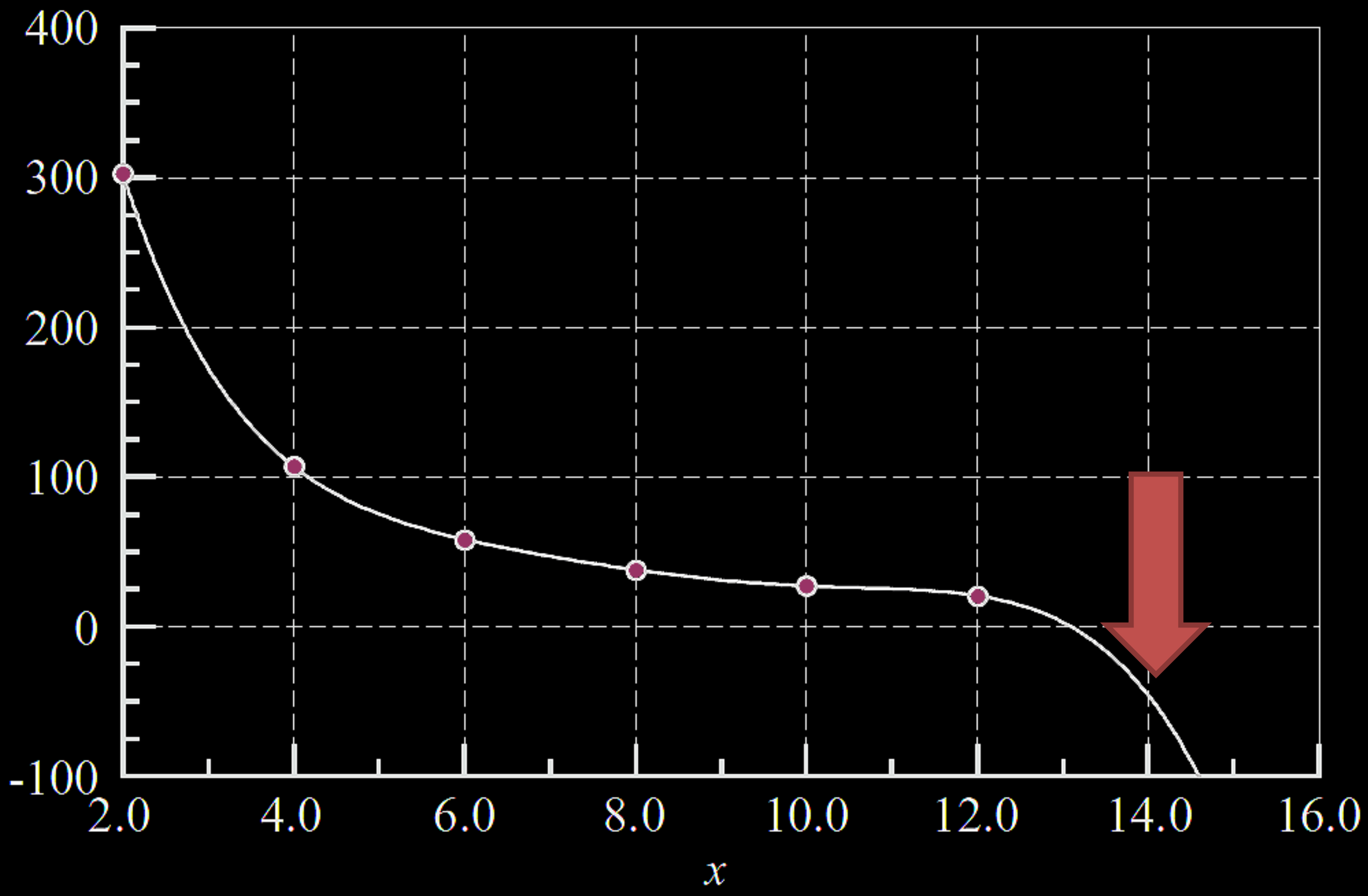
Polynomial Interpolation

Polynomial interpolant displaying oscillations



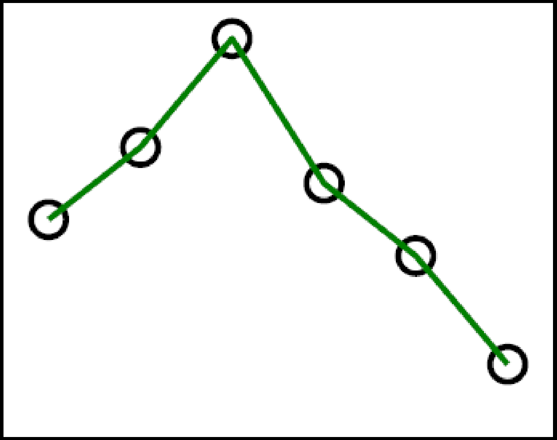
Polynomial Interpolation

Extrapolation may not follow the trend of data

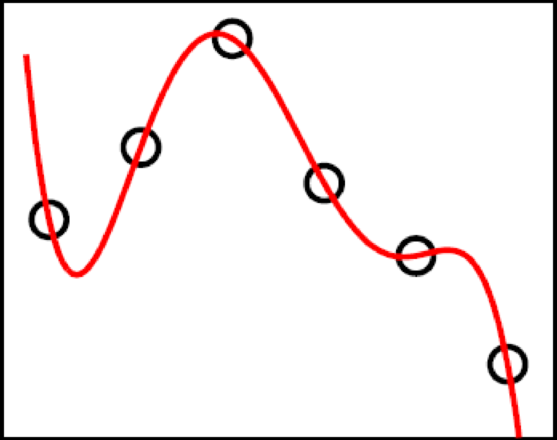


Interpolation

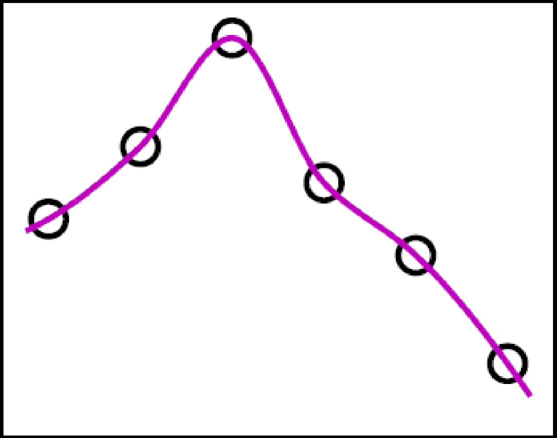
Piecewise linear interpolation



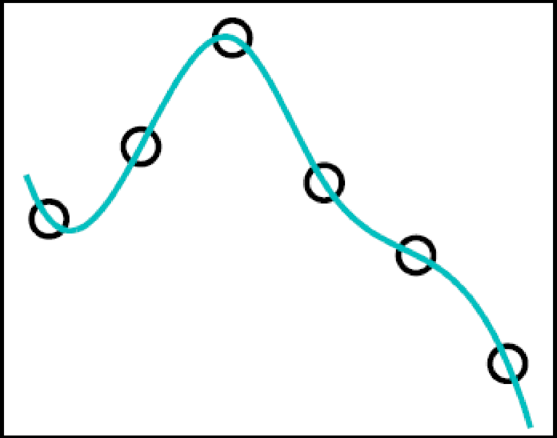
Full degree polynomial interpolation



Shape−preserving Hermite interpolation



Spline interpolation



Polynomial Fitting and Interpolation

Extrapolation may not follow the trend of data

```
numpy.interp(x, xp, fp, left=None, right=None, period=None)
```

One-dimensional linear interpolation.

```
>>> import numpy as np
>>> xp = [1, 2, 3]
>>> fp = [3, 2, 0]
>>> np.interp(2.5, xp, fp)
1.0
>>> np.interp([0, 1, 1.5, 2.72, 3.14], xp, fp)
array([3.  , 3.  , 2.5 , 0.56, 0.  ])
```

Polynomial Fitting and Interpolation

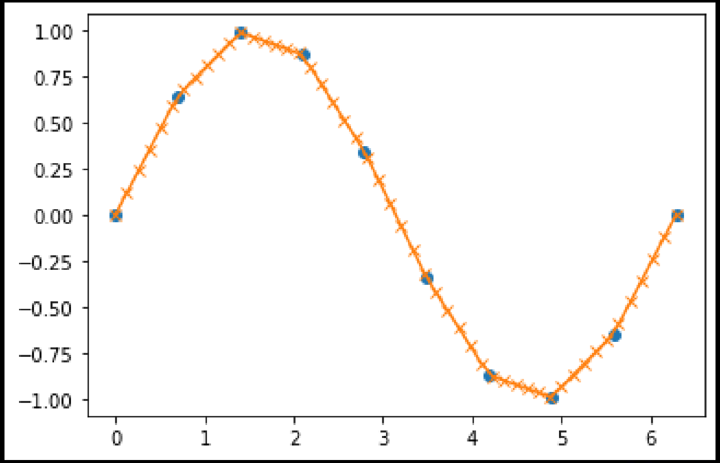
```
import numpy as np
import matplotlib.pyplot as plt

x = np.linspace(0, 2*np.pi, 10)
y = np.sin(x)

xvals = np.linspace(0, 2*np.pi, 50)
yinterp = np.interp(xvals, x, y)

fig = plt.figure()
plt.plot(x, y, 'o')
plt.plot(xvals, yinterp, '-x')

plt.show()
```



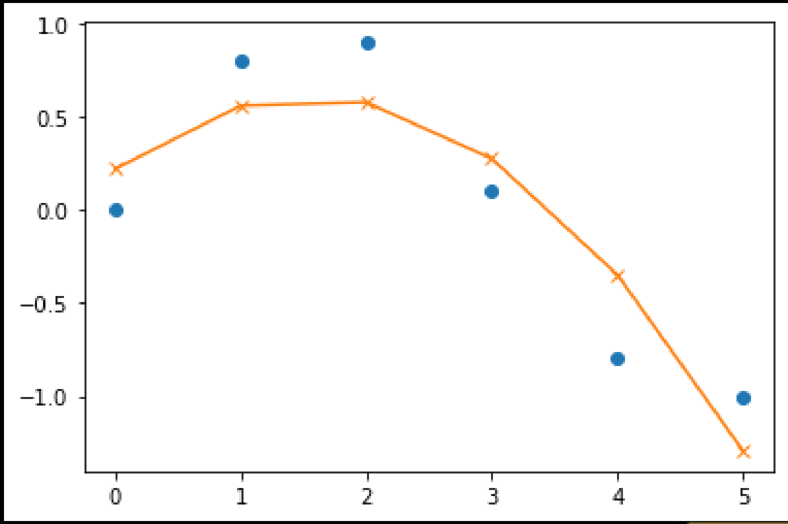
Polynomial Fitting and Interpolation

```
import numpy as np
import numpy.polynomial.polynomial as poly
import matplotlib.pyplot as plt

x = np.array([0.0, 1.0, 2.0, 3.0,  4.0,  5.0])
y = np.array([0.0, 0.8, 0.9, 0.1, -0.8, -1.0])

coefs = poly.polyfit(x, y, 2)
yfit = poly.polyval(x, coefs)

fig = plt.figure()
plt.plot(x, y, 'o')
plt.plot(x, yfit, '-x')
plt.show()
```



Polynomial Fitting and Interpolation

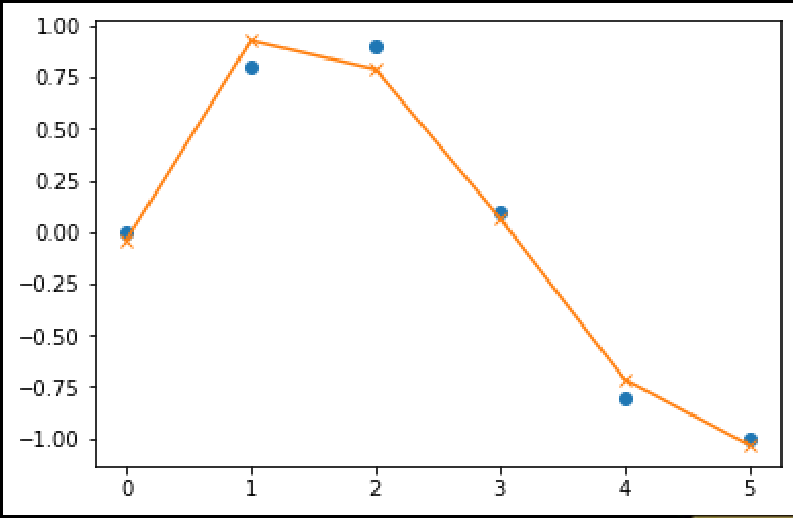
NumPy

```
import numpy as np
import numpy.polynomial.polynomial as poly
import matplotlib.pyplot as plt

x = np.array([0.0, 1.0, 2.0, 3.0,  4.0,  5.0])
y = np.array([0.0, 0.8, 0.9, 0.1, -0.8, -1.0])

coefs = poly.polyfit(x, y, 3)
yfit = poly.polyval(x, coefs)

fig = plt.figure()
plt.plot(x, y, 'o')
plt.plot(x, yfit, '-x')
plt.show()
```



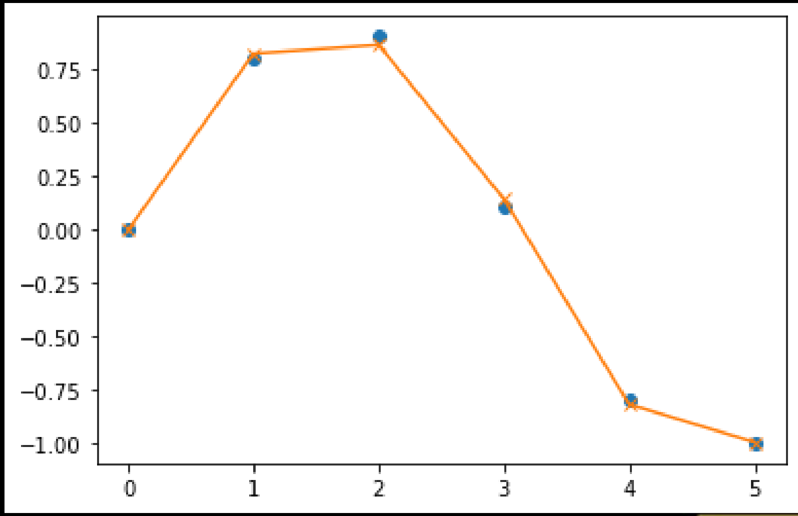
Polynomial Fitting and Interpolation

```
import numpy as np
import numpy.polynomial.polynomial as poly
import matplotlib.pyplot as plt

x = np.array([0.0, 1.0, 2.0, 3.0,  4.0,  5.0])
y = np.array([0.0, 0.8, 0.9, 0.1, -0.8, -1.0])

coefs = poly.polyfit(x, y, 4)
yfit = poly.polyval(x, coefs)

fig = plt.figure()
plt.plot(x, y, 'o')
plt.plot(x, yfit, '-x')
plt.show()
```



Interpolation

```
y = np.interp(x, xData, yData)
c = polyfit(xData, yData, m)
y = polyval(x, c)
```

Hız	kah	laktat
0		
8	134	2.10
9	150	2.20
10	160	2.60
11	170	3.20
12	175	3.80
13	178	4.50
14	185	5.40
15	188	7.30
16	192	10.00

7.5
8
8.5
9
9.5
10
10.5
11
11.5
12
12.5
13
13.5
14
14.5
15
15.5
16
16.5

?

Interpolation

```
y = np.interp(x, xData, yData)
c = polyfit(xData, yData, m)
y = polyval(x, c)
```

Hız	kah	laktat	VO2 (ml/kg/dk)
0			
8	133	1.18	22.90
9	132	1.46	25.17
10	144	1.68	26.68
11	151	1.76	31.22
12	152	2.10	31.29
13	160	2.61	34.97
14	169	2.92	37.77
15	174	4.58	39.49
16	178	6.00	41.75

7.5
8
8.5
9
9.5
10
10.5
11
11.5
12
12.5
13
13.5
14
14.5
15
15.5
16
16.5

?

Mid-term Exam

Bulls & Cows

Number Bulls & Cows
History

This traditional pencil and paper game probably dates back to the 1900s.

Bulls and Cows is a popular game for programming on a computer. The earliest recorded program, called MOO, written in 1970 by J. M. Grochow at MIT in the PL/I computer language for the Multics by a similar program written by Frank King in 1968 and running on the Cambridge University mainframe.

In 1970 a version of the game was designed using colored pegs, and marketed by Invicta Plastics as "Mastermind".

Mid-term Exam

Bulls & Cows

One player, the Chooser, thinks of a four-digit number and the other player, the Guesser, tries to guess it. At each turn the Guesser tries a four digit number, and the Chooser says how close it is to the answer by giving:

The number of Bulls - digits correct in the right position.
The number of Cows - digits correct but in the wrong position.

The Guesser tries to guess the answer in the fewest number of turns. If either number has repeated digits the rule is that each digit can only count towards the score once, and Bulls are counted before Cows.
For example, if the Chooser has thought of the number 2745 the replies for some guesses are as follows:

Guess: 1389 - Reply: 0 Bulls, 0 Cows.
Guess: 1234 - Reply: 0 Bulls, 2 Cows.
Guess: 1759 - Reply: 1 Bull, 1 Cow.
Guess: 1785 - Reply: 2 Bulls, 0 Cows.
Guess: 2745 - Reply: 4 Bulls!

Mid-term Exam

Write a Mastermind Game#1

Mastermind or Master Mind is a code-breaking game for two players. The modern game with pegs was invented in 1970 by Mordecai Meirowitz, an Israeli postmaster and telecommunications expert. It may have been inspired by moo, a program developed by J. M. Grochow at MIT in the late 1960s. In the game, one player is the codemaker and the other is the codebreaker.

The codemaker secretly selects a code consisting of an ordered sequence of four colors `(`c_1`,` c_2`,` c_3`,` c_4`)`, each chosen from a set of six possible colors, with repetitions allowed. The codebreaker then tries to guess the code by repeatedly proposing a sequence of colors . After each guess, the codemaker tells the codebreaker two numbers `(b, w)`: the number of correct colors in the correct positions (`b`) and the number of colors that are part of the code but not in the correct positions (`w`). For example, if the code is `(1,2,3,3)` and the codebreaker's guess is `(3,2,4,3)`, then the codemaker's response would be `(2,1)`, since the codebreaker has guessed the 2 and the second 3 correctly and in the correct positions, while having guessed the first three correctly but in the wrong position.

Mid-term Exam

The codebreaker continues guessing until he guesses the code correctly or until he reaches a maximum allowable number of guesses without having correctly identified the secret code. Interestingly, can be computed as

$$w = \left(\sum_{i=1}^{6} min(c_i, g_i) \right) - b$$

Where c_i is the number of times the color i is in the code and g_i is the number of times it is in the guess.

Knuth (1976-77) showed that the codebreaker can always succeed in five or fewer moves (i.e., knows the code after four guesses). His technique uses a greedy strategy that minimizes the number of remaining possibilities at each step, and requires 4.478 guesses on average, assuming equally likely code choice. Irving (1978-79) subsequently found a strategy with slightly smaller average length. Koyama and Lai (1993) described a strategy that minimizes the average number of guesses, requiring on average 4.340 guesses, although may require up to six in the worst case. A slight modification also described by Koyama and Lai (1993) increases the average to 4.341, but reduces the maximum number of guesses required to five.

Mid-term Exam

The "static" problem of finding the minimum number of guesses the codebreaker can make all at once at the beginning of the game without waiting for the answers, and then upon receiving the answers, completely determine the code in the next "guess" (Chvatal 1983), can be solved with six initial guesses (Greenwell 1999-2000). One particular combination that allows the codebreaker to know the code after six guesses (and so require a seventh to reveal his knowledge of the solution) is (1, 2, 2, 1), (2, 3, 5, 4), (3, 3, 1, 1), (4, 5, 2, 4), (5, 6, 5, 6), (6, 6, 4, 3). It is not known if this number can be reduced to five (exhaustive checking would require computations), although it is believed not.

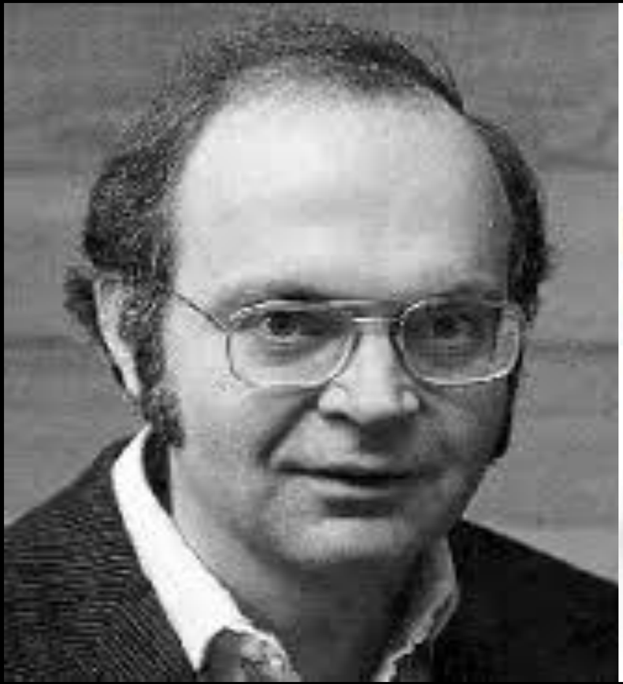
Swaszek (1999-2000) gives an analysis of practical strategies that do not require complicated record-keeping or use of a computer. Making a random guess from the set of remaining candidate code sequences gives a surprisingly short average game length of 4.638, while interpreting each guess as a number and using the next higher number consistent with the known information gives a game of average length 4.758.

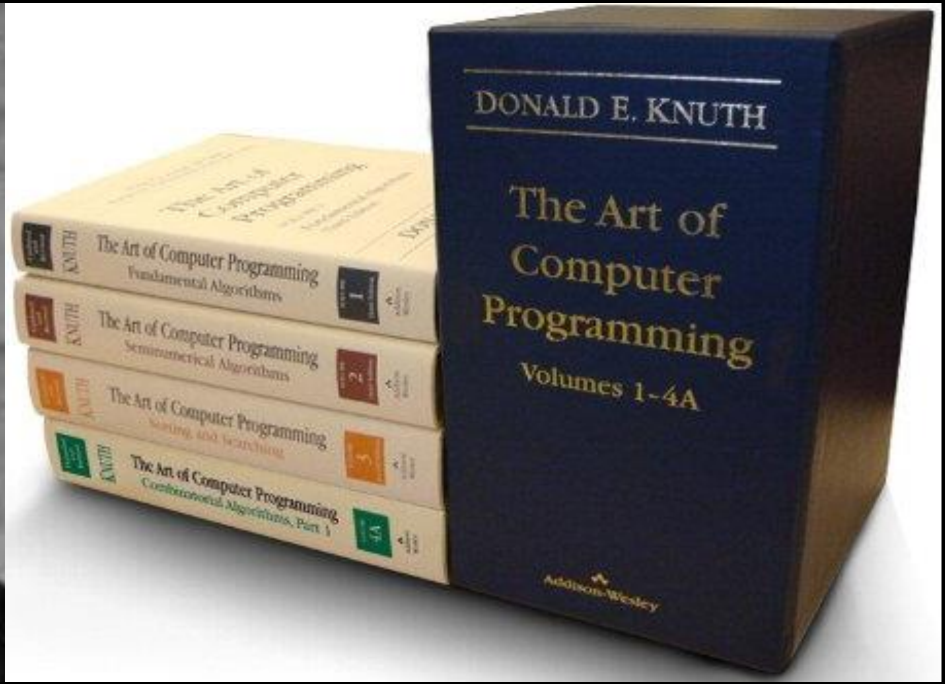
Chvatal, V. "Mastermind." *Combinatorica* **3**, 325-329, 1983.
Greenwell, D. L. "Mastermind." *J. Recr. Math.* **30**, 191-192, 1999-2000.
Irving, R. W. "Towards an Optimum Mastermind Strategy." *J. Recr. Math.* **11**, 81-87, 1978-79.
Knuth, D. E. "The Computer as a Master Mind." *J. Recr. Math.* **9**, 1-6, 1976-77.
Koyama, K. and Lai, T. W. "An Optimal Mastermind Strategy." *J. Recr. Math.* **25**, 251-256, 1993.
Swaszek, P. F. "The Mastermind Novice." *J. Recr. Math.* **30**, 193-198, 1999-2000.

Mid-term Exam

```
Which one you are? [default] codebreaker or [1] codemaker : <ENTER>
1) Code Breaker will guess the CODE assumed by Code Maker ',...
2) The Hint provided by Code Maker should provide the ',...
   information of Blacks & Whites',...
3) BLACK represents Correct Color in Correct Position.',...
4) WHITE represents Correct Color in Wrong Position.',...
NOTE: The COLOR that is represented as BLACK cannot be used to represent
as WHITE.',...
All the colors in a guess should be distinct.'...
Always the clue of BLACK appended with WHITE should be given not the vice-
versa.
>> Guess my combination ? cbrg <ENTER>
>> your 1. guess is (c)yan (b)lue (r)ed (g)reen
>> you have got 1 (b)lack and 2 (w)hite
……
……
>> your 4. guess is (c)yan (r)ed (b)lue (y)ellow
>> you have got 4 (b)lack, you broke the code!!

>> Which one you are? [default] codebreaker or [1] codemaker : 1 <ENTER>
>> Please pick a combination and take a note press <ENTER> to start
>> my 1. guess is (c)yan (b)lue (r)ed (g)reen
>> please enter how many black and white I have got : 0, 1
>> my 2. guess is (c)yan (m)agenta (y)ellow (g)reen
```




DONALD E. KNUTH
The Art of Computer Programming
Volumes 1-4A
Addison-Wesley
The Art of Computer Programming Fundamental Algorithms
The Art of Computer Programming Seminumerical Algorithms
The Art of Computer Programming Sorting and Searching
The Art of Computer Programming Combinatorial Algorithms, Part 1

Donald Ervin Knuth born January 10, 1938) is an American computer scientist, mathematician, and Professor Emeritus at Stanford University. He is the author of the multi-volume work The Art of Computer Programming. Knuth has been called the "father" of the analysis of algorithms. He contributed to the development of the rigorous analysis of the computational complexity of algorithms and systematized formal mathematical techniques for it.

Ara-Sınav

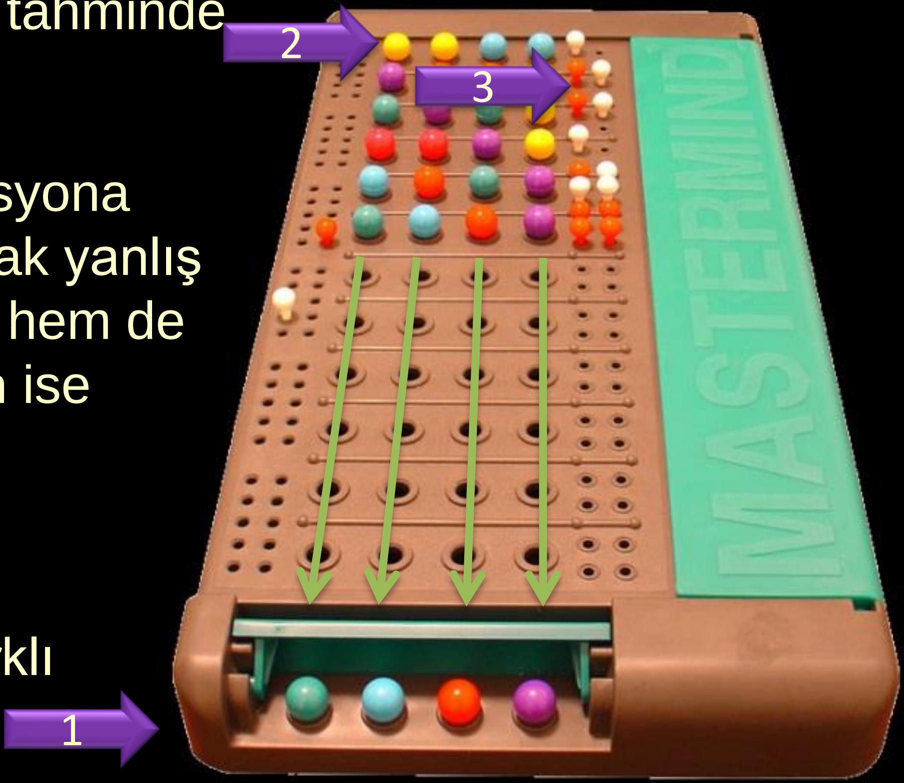
Mastermind, 1970'lerin başında Mordecai Meirowitz tarafından geliştirilmiş, karşılıklı iki kişiyle oynanan bir masa oyunudur. Oyunculardan ilki kod yapıcı (codemaker) olarak adlandırılır ve görevi 6 elemanlı bir renk kümesinden 4 adet renk seçip, diğer oyuncudan çözmesini bekleyeceği gizli bir renk kombinasyonu oluşturmaktır. İkinci oyuncu ise kod kırıcı (codebreaker) olarak adlandırılır ve amacı ilk oyuncunun oluşturduğu renk kombinasyonunu en fazla 13 aşamada bulmaktır.

Ara-Sınav

2. Daha sonra kod kırıcı, rastgele bir tahminde bulunur.

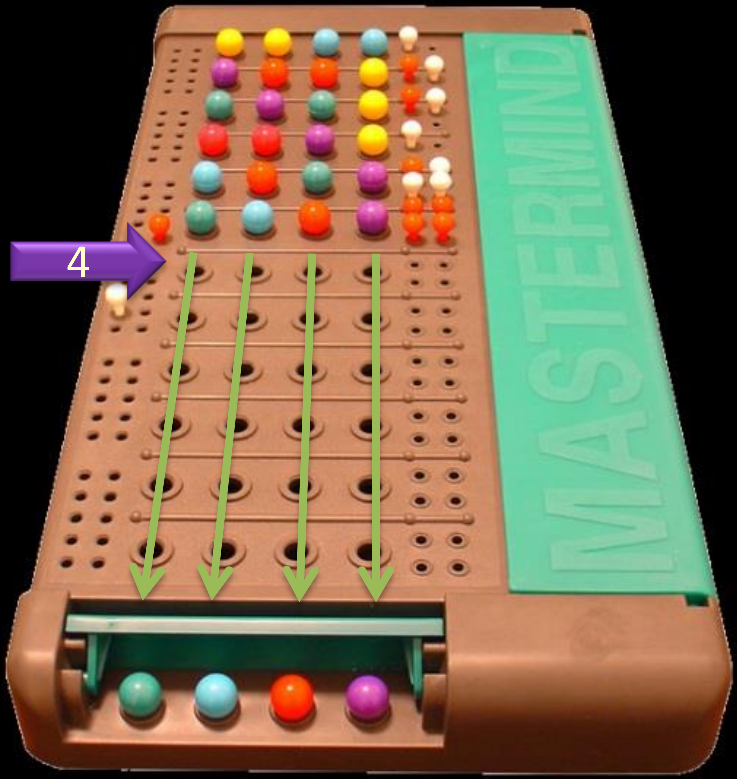
3. Tahmin neticesinde, gizli kombinasyona göre renk olarak doğru fakat yer olarak yanlış her boncuk için beyaz çivi, hem renk hem de yer olarak doğru olan her boncuk için ise turuncu çivi yerleştirilir.

1. İlk oyuncu 6'lık bir kümeden diğer oyuncunun görmeyeceği şekilde 4 farklı renkte boncuk(color peg) seçip gizli bölmeye dizer.



Ara-Sınav

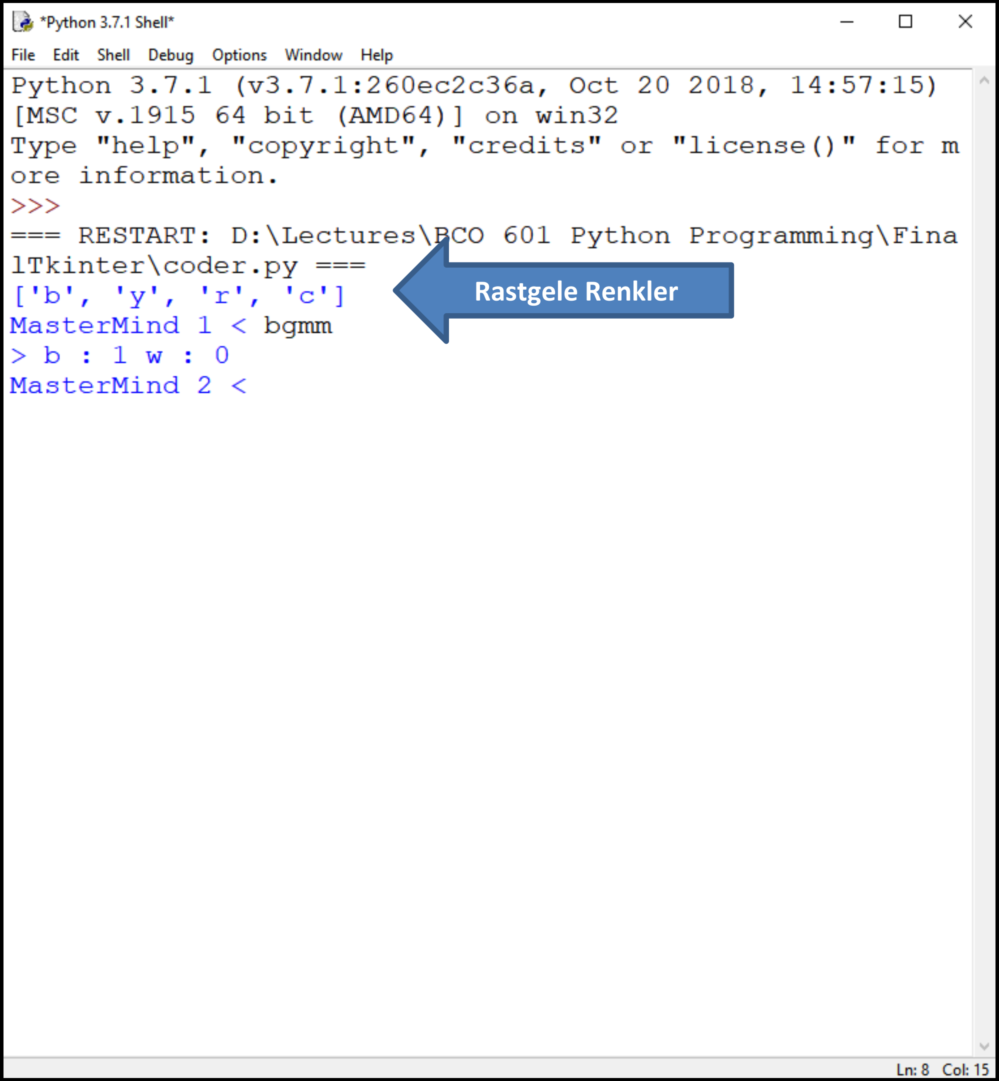
4. Turuncu ve beyaz çiviler çözüm kümesini kısıtlayarak kodu çözmeye çalışan oyuncuyu doğru cevaba yönlendirir. Oyunun amacı en fazla 13 tahminde doğru cevaba ulaşmaktır.



Ara-Sınav

Yazacağınız program 6 farklı bir renk kümesinden rastgele 4 renk seceçektir. Seçilen renk kombinasyonda <u>bir renk en fazla iki kez</u> kullanılabilir. Oyuncu, bir tahminde bulunur. Tahmin neticesinde, gizli kombinasyona göre renk olarak doğru fakat yer olarak yanlış her renk için **beyaz** , hem renk hem de yer olarak doğru olan her renk için ise **siyah** bilgisi verecek. Yazdığınız program; oyuncunun girdiği renk sayısını, renk bilgisini kontrol ederek oyuncuya bilgi verecektir.

Ara-Sınav

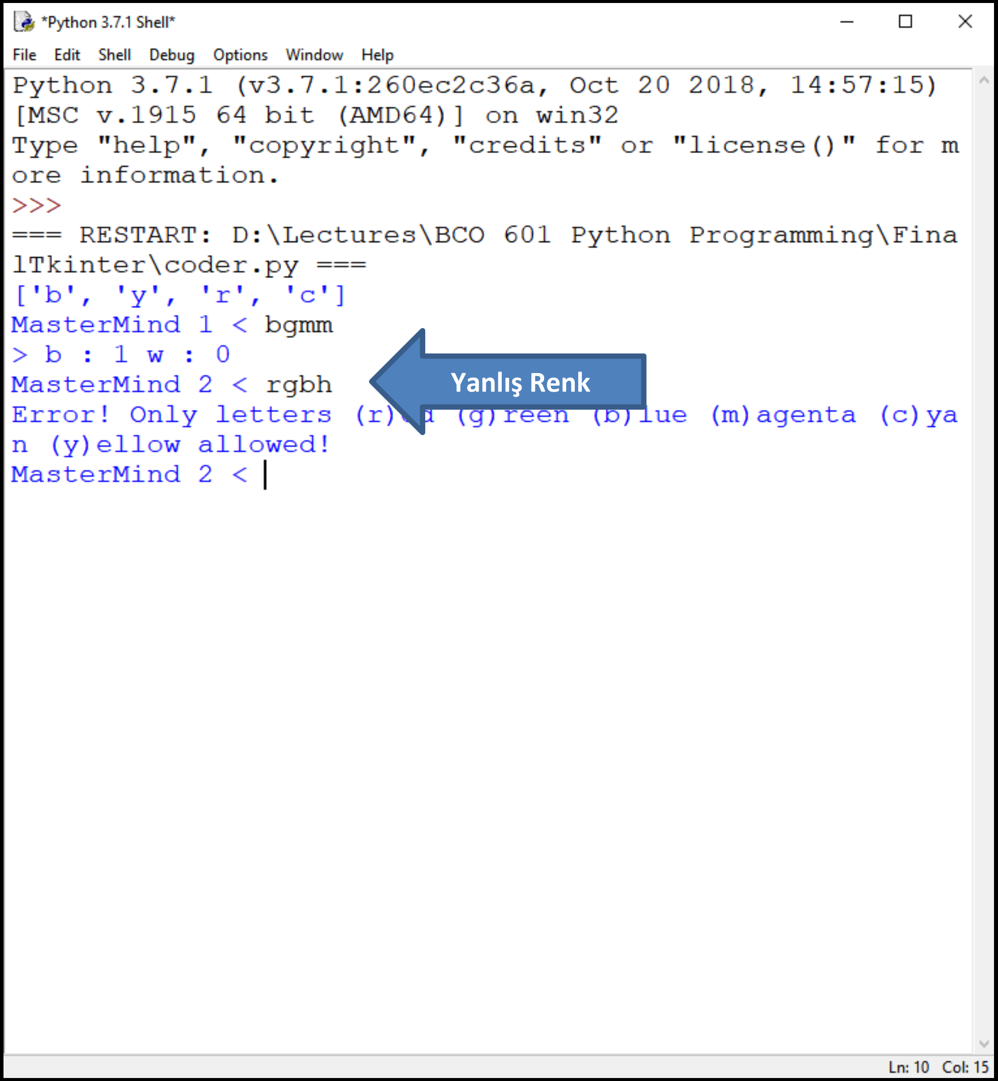


```
Python 3.7.1 (v3.7.1:260ec2c36a, Oct 20 2018, 14:57:15)
[MSC v.1915 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for m
ore information.
>>>
=== RESTART: D:\Lectures\BCO 601 Python Programming\Fina
lTkinter\coder.py ===
['b', 'y', 'r', 'c']
MasterMind 1 < bgmm
> b : 1 w : 0
MasterMind 2 <
```

Oyuncunun Tahmini

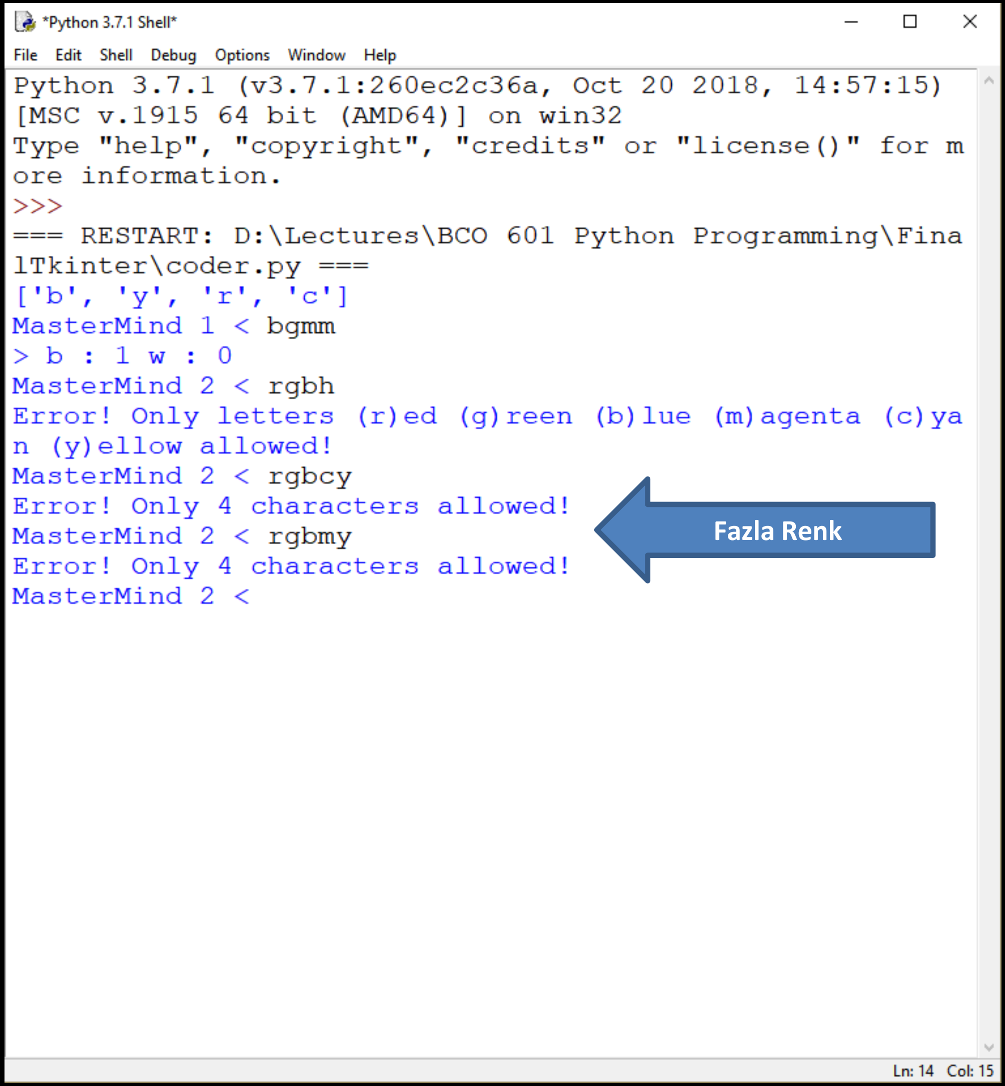
Rastgele Renkler

Ara-Sınav



```
Python 3.7.1 (v3.7.1:260ec2c36a, Oct 20 2018, 14:57:15)
[MSC v.1915 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for m
ore information.
>>>
=== RESTART: D:\Lectures\BCO 601 Python Programming\Fina
lTkinter\coder.py ===
['b', 'y', 'r', 'c']
MasterMind 1 < bgmm
> b : 1 w : 0
MasterMind 2 < rgbh
Error! Only letters (r)ed (g)reen (b)lue (m)agenta (c)ya
n (y)ellow allowed!
MasterMind 2 <
```

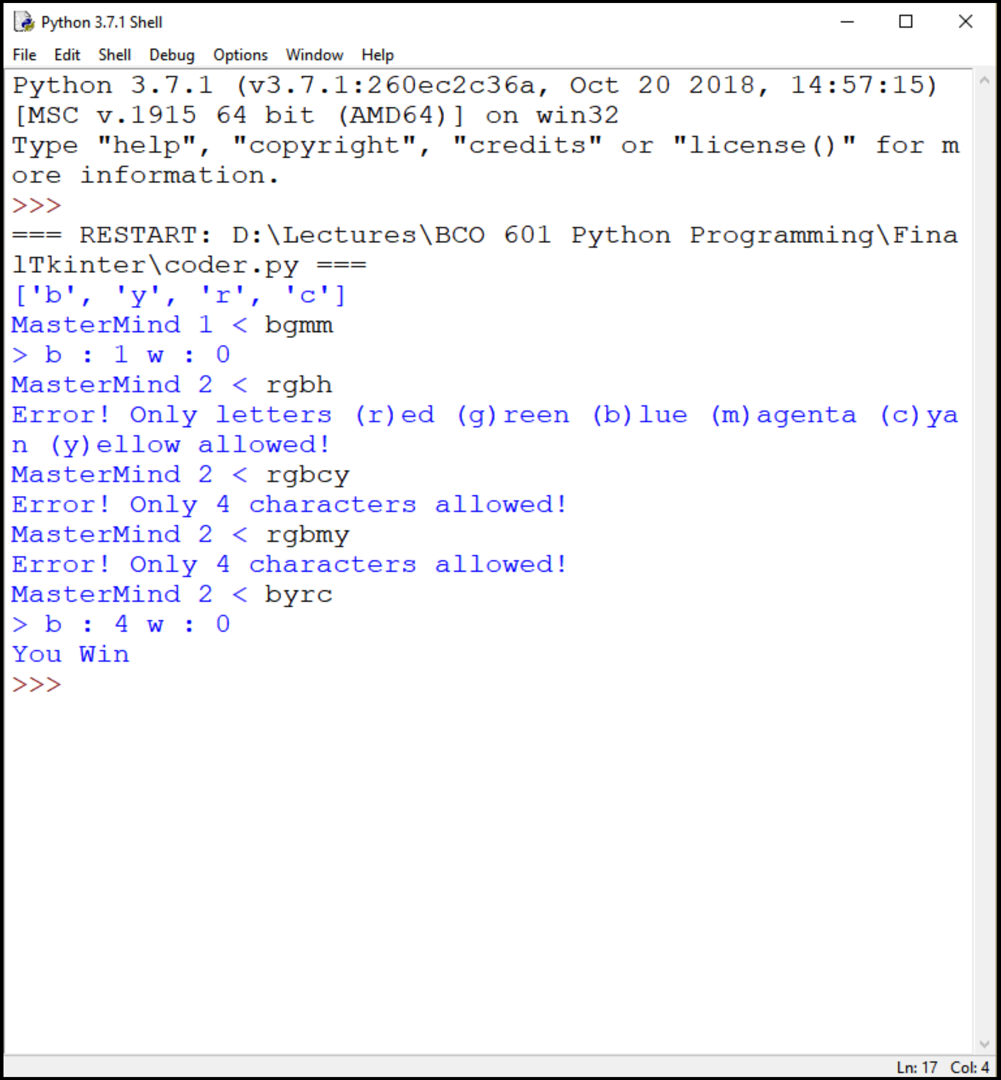
Ara-Sınav



```
Python 3.7.1 (v3.7.1:260ec2c36a, Oct 20 2018, 14:57:15)
[MSC v.1915 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for m
ore information.
>>>
=== RESTART: D:\Lectures\BCO 601 Python Programming\Fina
lTkinter\coder.py ===
['b', 'y', 'r', 'c']
MasterMind 1 < bgmm
> b : 1 w : 0
MasterMind 2 < rgbh
Error! Only letters (r)ed (g)reen (b)lue (m)agenta (c)ya
n (y)ellow allowed!
MasterMind 2 < rgbcy
Error! Only 4 characters allowed!
MasterMind 2 < rgbmy
Error! Only 4 characters allowed!
MasterMind 2 <
```

Fazla Renk

Ara-Sınav



```
Python 3.7.1 (v3.7.1:260ec2c36a, Oct 20 2018, 14:57:15)
[MSC v.1915 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for m
ore information.
>>>
=== RESTART: D:\Lectures\BCO 601 Python Programming\Fina
lTkinter\coder.py ===
['b', 'y', 'r', 'c']
MasterMind 1 < bgmm
> b : 1 w : 0
MasterMind 2 < rgbh
Error! Only letters (r)ed (g)reen (b)lue (m)agenta (c)ya
n (y)ellow allowed!
MasterMind 2 < rgbcy
Error! Only 4 characters allowed!
MasterMind 2 < rgbmy
Error! Only 4 characters allowed!
MasterMind 2 < byrc
> b : 4 w : 0
You Win
>>>
```