



Crack The Code

Crack the Code is a code breaking game. Each time you play the game, a 5 number code is generated. The sum for the five numbers and the number of odds is displayed on the screen. Each time you guess an integer, the program searches through the secret code for a match. If a match is found, that space is revealed on your guess code. Once you have guessed all the numbers correctly, the code is displayed as well as the number of guesses it took to break the code.

Objectives:

Programming Objectives:

- Use appropriate libraries
- Create random integers
- Create procedures to modularize code
- Create iterative statements
- Create appropriate conditional statements
- Use modular arithmetic
- Use lists to store and manage data

Key AP Computer Science Principles Standards:

- Represent a value with a variable (AAP-1.A)
- Represent a list or string using a variable (AAP-1.C)
- Develop data abstraction using lists to store multiple elements (AAP-1.D)
- Write expressions using logical operators (AAP-1.D)
- Write conditional statements (AAP-2.H)
- Write iteration statements (AAP-2.K)
- Write expressions to generate possible values (AAP-3.E)
- Write expressions that use list indexing and list procedures (AAP-2.N)
- Write iteration statements to traverse a list (AAP-2.O)
- Write statements to call procedures (AAP-3.A)
- Develop procedural abstractions to manage complexity in a program by writing procedures (AAP-3.C)
- Select appropriate libraries or existing code segments to use in creating new programs (AAP-3.D)

In this project, you will create a secret code game. Your code will generate a secret 5-digit code. It will display the sum for the 5 numbers and print the number of odd integers in the code. Initially, the user will have an empty code list of `['_','_','_','_','_']`. As the user guesses correctly, the underscores, `_`, will be replaced with the correct integers. Once the user guesses all 5 numbers, your game will display the number of guesses it took to crack the code.

Sample Game:

```

1.1 1.2 1.3 *Crack T. ode RAD 5/5
Python Shell
>>>sum 10
# odds 2
['_','_','_','_','_']
>>>
guess:

```

Original Question

```

1.1 1.2 1.3 *Crack T. ode RAD 11/11
Python Shell
>>>sum 10
# odds 2
['_','_','_','_','_']
>>>
guess: 5
>>>
sum 10
# odds 2
['_','_','_','_','_']
>>>
guess:

```

User guessed a 5. It isn't in the code list. Asks for a new guess. Adds 1 to guesses.

```

1.1 1.2 1.3 *Crack T. ode RAD 17/17
Python Shell
sum 10
# odds 2
['_','_','_','_','_']
>>>
guess: 1
>>>
sum 10
# odds 2
['_','_','_','_','_']
>>>
guess:

```

User guessed a 1. It is in the code list. Show the 1 location. Add 1 to the total guesses

.....

```

1.1 1.2 1.3 *Crack T. ode RAD 39/39
Python Shell
guess: 2
>>>
sum 10
# odds 2
[2, 1, 0, 4, 3]
>>>
guess: 4
>>>
code broken in 6 guesses
[2, 1, 0, 4, 3]
>>>
guess:

```

After 4 more guesses, the secret code is broken. Display the number of total guesses.



1. In this project, you will generate random integers between 0 and 9. You will need to clear the shell. What library imports should you include?

Include those library imports at the top of your project.

For simplicity, we will start with 0-9. After you have completed the project, you may adjust this to any range of your choice.

2. Write pseudocode for a **function** to generate a random list of 5 integers, each integer between 0 and 9. Determine and store the number of odd numbers in the list as well as the sum of the list. Return the code list with 5 integers, the sum and the number of odd integers. Call your function newCode.

Things to consider:

How do you randomly generate integers from 1 to 9?

How do you repeatedly add items to a list?

How do you determine if an integer is odd?

How do you use accumulators to keep track of a sum?

How do you return values from a function?



3. Code your function.

To verify your function works, make a **function call** and store the code list, sum, and number of odds. Print the code list, sum and number of odds.

Run your program several times. Make sure it generates new random numbers each time. Ensure the sum for the list and the number of odds in the list is correct.

4. Next, write the pseudocode to create a function named **check** that has three parameters; random number, code list, user guess list. First, make sure the user has entered a valid integer between 0 and 9. While this number is invalid, ask the user for a new number between 0 and 9. Search through each item in the code list. If the user's number matches the item in the list, replace the ___ in the guess list at that position with the user's number. Return the user's guess list.

Things to consider:

Make sure the code ensures the user entered a valid guess. As long as the guess is incorrect, ask for a new guess.

CRACK THE CODE STUDENT DOCUMENT

```
from random import *
from ti_system import *

def newCode():
    """
    Type Your Code Here
    """

code,sum, odds = newCode()
print(code)
print("sum",sum)
print("num odds", odds)
```

Sample Run

```
Python Shell 6/6
>>>#Running CrackTheCode1.py
>>>from CrackTheCode1 import *
[6, 4, 5, 6, 7]
sum 28
num odds 2
>>>
```



5. Code your function.

Pass the guess, test code list, and the empty guess list to the function. Print the returned list.

Execute your program. Make sure it matches the output to the right.

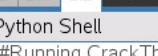
6. Change your sample guess line $n = 3$ to $n = 13$. Run your program.

Does it prompt you to enter a correct value?

Enter another incorrect value. Does it again prompt you to enter a value between 0 and 9?

Finally, enter a 1 for a guess. Verify your code replaces the two 1's at the end of your guess list. (Remember, the sample screenshot on step 5 passed the list [5,3,4,1,1]. Therefore, the sample code at the right has two 1's at the end.)

```
Python Shell 4/4
>>>#Running CrackTheCode1.py
>>>from CrackTheCode1 import *
['_', 3, '_', '_', '_']
>>>
```



Python Shell 7/7

```
>>>#Running CrackTheCode1.py
>>>from CrackTheCode1 import *
must be 0-9: 13
must be 0-9: 15
must be 0-9: 1
['_', '_', '_', '1', '1']
>>>
```



Computer Science with Python

TI-NSPIRE™ CX II TECHNOLOGY

7. Now that you have two properly working functions, remove the trial print statements and test data from step 5.

CRACK THE CODE

STUDENT DOCUMENT

```
from random import *
from ti_system import *

def newCode():
    Coded in Step 3

def check(n,code,guesses):
    Coded in Step 5

code,sum, odds = newCode()
```

8. Create an accumulator variable to store the total number of guesses.

Create a guess array with 5 empty “_”.

As the user guesses correct numbers, the “_” will change to the correct value.

Clear the python shell.

```
from random import *
from ti_system import *

def newCode():
    Coded in Step 3

def check(n,code,guesses):
    Coded in Step 5

code,sum, odds = newCode()

#set accumulator to 0
#create guess array with 5 "_"

Add Code Here

#clear the screen

Add Code Here
```



Computer Science with Python

TI-NSPIRE™ CX II TECHNOLOGY

9. Now, you'll need to write the code to:
Display the important information about the code list and the guess list. Ask the user for a guess number. Check the entered number and update both the guess list and the accumulator variable. Repeat this pattern until the guess code is correct.

Write your pseudocode in the space below.

The screen shots on the right are meant to help you think about the lines of code you need to include.

CRACK THE CODE

STUDENT DOCUMENT

Sample initial screen

```
Python Shell 5/5
>>>sum 10
# odds 2
['_','_','_','_','_','_']
>>>
guess:
```

Screen after user enters a 5.

5 is not inside the secret code.

```
Python Shell 11/11
>>>sum 10
# odds 2
['_','_','_','_','_','_']
>>>
guess: 5
>>>
sum 10
# odds 2
['_','_','_','_','_','_']
>>>
guess:
```

Screen after user enters a 1.

1 appears once in the secret code.

```
Python Shell 17/17
sum 10
# odds 2
['_','_','_','_','_','_']
>>>
guess: 1
>>>
sum 10
# odds 2
['_','_','1','_','_','_']
>>>
guess: |
```



Computer Science with Python

TI-NSPIRE™ CX II TECHNOLOGY

10. Write the code for your pseudo code above.

CRACK THE CODE

STUDENT DOCUMENT

The screenshot shows a TI-NSPIRE CX II calculator screen with a Python program titled 'CrackTheCode1.py'. The program is written in Python 3. The code includes imports for 'random' and 'ti_system', a function 'newCode()' that generates a random code and sum, a function 'check(n, code, guesses)' that checks guesses, and a main loop that repeatedly asks for a number until the code is guessed. The code is partially filled with boxes indicating where to add code from previous steps.

```
1.3 2.1 2.2 *CrackT_e_2 RAD 1/32
CrackTheCode1.py
from random import *
from ti_system import *

def newCode():
    Coded in Step 3

def check(n,code,guesses):
    Coded in Step 5

code,sum, odds = newCode()

#set accumulator to 0
#create guess array with 5 "_"
Coded in Step 8

#clear the screen
Coded in Step 8

#Loop repeatedly asks for a number
#updates the guess list
#Repeats until the secret code is guessed
Add Code Here
```

11. Run your program.

- ☐ Verify the loop stops when the secret code is guessed.
- ☐ Make sure the integers in the code add up to the displayed sum
- ☐ Verify the number of odds in the code matches the odd count.



Computer Science with Python

TI-NSPIRE™ CX II TECHNOLOGY

12. Add a print statement that will tell the user the code has been broken. Tell the user how many guesses it took to break the code. Display the final code.

Run your code several times.

Verify the following

- ☐ Each time the user plays the game a new code is generated.
- ☐ Once the code has been guessed, the number of guesses is displayed and correct.

.

.

A sample screenshot is provided to the right for such a trial.

CRACK THE CODE

STUDENT DOCUMENT

Initial Screen:

```
Python Shell 5/5
>>>sum 30
# odds 2
['_','_','_','_','_','_']
>>>
guess:|
```

.....

.....

Display after secret code is broken

```
Python Shell 45/45
guess: 8
>>>
sum 30
# odds 2
[7, 8, '_', '_', '_', 3]
>>>
guess: 6
>>>
code broken in 7 guesses
[7, 8, 6, 6, 3]
>>>|
```