Python Exercises 2b: Repetition Control Structures *Solutions*

Table of Contents

Python Exercises 2b: Repetition Control Structures *Solutions*

1.Dice Game: Sixes Bet

Sixes Bet is an old dice game. The bet is simple: will the player roll a 6 at least once out of four rolls?

Write a Python program to play the game. The program should allow the player to roll the die up to 4 times, display the roll, and check if the player has rolled a six. If the player has rolled a 6, the game is over and the player has won. If the player rolls 4 times and hasn't rolled a 6, s/he loses.

Specification Table

Input	Processing	Output

	Set count to 0
	While count < 4
	  Roll the die
	  Display number rolled
	  If number rolled = 6
	  Display “you win”
	  break from while loop
	  Increase count by 1
	Else
	  Display “You lose”
		die roll
		message

Python Program

Version 1: While Loop

```
# Program Name: section02_solutions10_sixesbet.py
# Purpose: Play the game of Sixes Bet
# EXAMPLE OF: counting while loop with break
from random import randint

count = 0 # number of dice rolls so far

#keep going until 4 dice have been rolled
while count < 4:
    #simulate die roll
    roll = randint(1,6)
    print("You rolled: ", roll)

    # did the player win?
    if roll == 6:
        print("You win")
        break # game over

    count = count + 1
# else is executed when while loop condition is false
else:
    print("You lose")
```

Version 2: for loop

```
# Program Name: section02_solutions10_sixesbet.py
# Purpose: Play the game of Sixes Bet
# EXAMPLE OF: counting for loop with break
from random import randint

#keep going until 4 dice have been rolled
for i in range(4): # range provides 4 numbers: 0, 1, 2, 3
    #simulate die roll
    roll = randint(1,6)
    print("You rolled: ", roll)

    # did the player win?
    if roll == 6:
        print("You win")
        break # game over
# else is executed when for loop condition is false
else:
    print("You lose")
```

Sample Output (Same for both versions)

Win on the first roll

```
You rolled:  6
You win
```

Win on the second roll

```
You rolled:  2
You rolled:  6
You win
```

Win on the third roll

```
You rolled:  5
You rolled:  4
You rolled:  6
You win
```

Win on the fourth roll

```
You rolled:  4
You rolled:  2
You rolled:  4
You rolled:  6
You win
```

Lose

```
You rolled:  4
You rolled:  4
You rolled:  5
You rolled:  1
You lose
```

2.Long Jump

In a long jump competition, a jumper takes 6 jumps, and his/her best jump is recorded.

Jump	**1**	**2**	**3**	**4**	**5**	**6**
Distance	8.37	8.43	8.40	8.11	8.31	8.44

Write a program which inputs 6 long jump distances displays the average, maximum and minimum jump.

Specification Table

Input	**Processing**	**Output**
distance	For each number i from 0 to 5 Input distance Add distance to total If i = 0 (first jump) Set maximum to 0 Set minimum to 0 Else If distance > maximum (new maximum) Set maximum to distance Else If distance < minimum (new minimum) Set minimum to distance Display maximum Display Minimum Calculate average Display average	average maximum minimum

Python Program

```
# Program Name: section02_solutions10_longjump_for.py
# Purpose: calculate average, maximum and minimum jumps
# Example of: counting for loop

total = 0

# keep going until we do all 6
for i in range(6):
    #Input long jump distance in metres
    distance = float(input("Enter distance jumped: "))

    # add to total
    total += distance

    # if this is the first jump
    if i == 0:
        # set it to be the maximum and minimum (so far)
        maximum = minimum = distance
    # otherwise, is this a new maximum?
    elif distance > maximum:
        maximum = distance
    # otherwise, is this a new minimum?
    elif distance < minimum:
        minimum = distance

# While loop is finished - display results
print() # blank line
print(f"Best jump: {maximum}m")
print(f"Worst jump: {minimum}m")
print(f"Average jump: {total/6:.2f}")
```

Sample Output

```
Enter distance jumped: 8.37

Enter distance jumped: 8.43

Enter distance jumped: 8.40

Enter distance jumped: 8.11

Enter distance jumped: 8.31

Enter distance jumped: 8.44

Best jump: 8.44m
Worst jump: 8.11m
Average jump: 8.34
```

3.Leap Years

A year is a *leap year* if it is divisible by 4, unless it is a century year (divisible by 100) that is not divisible by 400. e.g. 1800 and 1900 were not leap years; 1600 and 2000 were. One way to approach this is: a year is a leap year

if it is divisible by 400
OR it is divisible by 4 and not divisible by 100

Write a program which inputs two years, and then determines and displays whether or not each year between them is a leap year.

To check if one number is evenly divisible by another, use the modulo operator, e.g.

```
if year % 4 == 0
```

checks if the value of `year` is evenly divisible by 4.

Test your program with 2000 to 2020, and then test it again with 1890 to 1910.

Specification Table

Input	**Processing**	**Output**
start end	For each year from start to end+1 Display year If year is evenly divisible by 400 or year is evenly divisible for 4 but not evenly divisible by 100 Display “Leap year Else Display “Not yeap year” Calculate average Display average	Leap year or Not leap year

Python Program

```
# Program Name: section02_solutions11_leapyear_for.py
# Purpose: determine leap years in a specified range
# Example of: counting for loop

# input the start and end years
start = int(input("Enter the first year: "))
end = int(input("Enter the last year: "))

# keep going until we get to the end
for year in range(start, end+1): # want to include end year, so use end+1
    print(year, end=": ")
    if year % 400 == 0 or (year % 4 == 0 and year % 100 != 0):
        print("Leap Year")
    else:
        print("Not a leap year")
```

Sample Output

```
Enter the first year: 2000

Enter the last year: 2020
2000: Leap Year
2001: Not a leap year
2002: Not a leap year
2003: Not a leap year
2004: Leap Year
2005: Not a leap year
2006: Not a leap year
2007: Not a leap year
2008: Leap Year
2009: Not a leap year
2010: Not a leap year
2011: Not a leap year
2012: Leap Year
2013: Not a leap year
2014: Not a leap year
2015: Not a leap year
2016: Leap Year
2017: Not a leap year
2018: Not a leap year
2019: Not a leap year
2020: Leap Year
```

```
Enter the first year: 1890

Enter the last year: 1910
1890: Not a leap year
1891: Not a leap year
1892: Leap Year
1893: Not a leap year
1894: Not a leap year
1895: Not a leap year
1896: Leap Year
1897: Not a leap year
1898: Not a leap year
1899: Not a leap year
1900: Not a leap year
1901: Not a leap year
1902: Not a leap year
1903: Not a leap year
1904: Leap Year
1905: Not a leap year
1906: Not a leap year
1907: Not a leap year
1908: Leap Year
1909: Not a leap year
1910: Not a leap year
```

4.Chess Reward

Legend has it that the game of chess was invented by Sissa ben Dahir of the court of King Shiram of India. The king was so impressed with this invention that he promised to give Sissa any reward he wished for.

Being a clever chap, Sissa gave the king a choice: the king could either give him 10,000 rupees, or a payment of wheat based on the 64 square chessboard. To pay Sissa in wheat, the king would need to give him one grain of wheat for the first square, 2 grains of wheat for the second square, 4 for the third, 8 for the forth, and so on. Each square should have double the amount of wheat as the previous square.

1	2	4	8	16	32	64	128
256	512						
							2^{63}

Now, the kingdom was known for its wheat production, so King Shiram was more than willing to part with some wheat. He thought Sissa a fool to ask for such an insignificant award.

Write a program which calculates and displays:

- the number of grains of wheat on each square,
- and the total number of grains of wheat.

Specification Table

Input	Processing	Output
	Set total to 0 For each number i from 0 to 63 (provided by range(64)) Set grains to 2^i (2 to the power of i, e.g. $2^0 = 1$, $2^1 = 2$, $2^2 = 4$) Display grains Add grains to toal	grains total

Python Program

```
# Program Name: section02_solutions13_chess_reward_for.py
# Purpose: To calculate Sissa ben Dahir's reward
# Exampe of: counting loop

# total grains of rice
total = 0

# loop to process the 64 squares of a chess board
for i in range(64):
    grains = 2 ** i # number of grains on current square
    print(grains, end=" ")
    total += grains

print()
print(f"Total: {total} grains of rice")
```

Sample Output

```
1 2 4 8 16 32 64 128 256 512 1024 2048 4096 8192 16384 32768 65536 131072 262144 524288 1048576 2097152 4194304 8388608 16777216
33554432 67108864 134217728 268435456 536870912 1073741824 2147483648 4294967296 8589934592 17179869184 34359738368 68719476736
137438953472 274877906944 549755813888 1099511627776 2199023255552 4398046511104 8796093022208 17592186044416 35184372088832
70368744177664 140737488355328 281474976710656 562949953421312 1125899906842624 2251799813685248 4503599627370496
9007199254740992 18014398509481984 36028797018963968 72057594037927936 144115188075855872 288230376151711744 576460752303423488
1152921504606846976 2305843009213693952 4611686018427387904 9223372036854775808
Total: 18446744073709551615 grains of rice
```

5.Guess the Number

Guess the Number is a simple guessing game. The computer "thinks" of a number, and the player has ten attempts to guess it. If the player guesses the number, s/he wins. Otherwise, the computer tells the player if the guess was too high or too low.

Design and write a program to play Guess the Number. The program randomly selects a number between 1 and 100, and then repeatedly (up to 10 times)

- prompts the player to input his/her guess
- checks if it is the right number, it should stop repeating
- otherwise displays a message indicating whether the guess was too high or too low

The program should then display a message indicating whether the player has won or lost.

Specification Table

Input	**Processing**	**Output**

guess	Generate a random number (using randint(1,100)
	For each number i from 0 to 9 (provided by range(10))
	  Input guess
	  If guess = number
	  Display "You win"
	  break from for loop
	  Else If guess < number
	  Display "Too low"
	  Else
	  Display "Too high"
	Else (for loop completed formally after 10 repetitions)
	  Display "Used all your guess – you lose"
		response to guess
		win/lose message

Python Program

```
# Program Name: section02_solutions14_guess.py
# Purpose: To play the Guess the Number game
# Exampe of: counting loop
from random import randint

# the computer "picks" a number
number = randint(1,100)

print("I'm thinking of a number between 1 and 100. Try to guess it.")

# the players has 10 guesses
for i in range(10):
    # input the player's guess
    guess = int(input("Enter your guess: "))

    # were they right?
    if guess == number:
        print("You guessed it!")
        break
    # or too low?
    elif guess < number:
        print("Too low, try again.")
    # otherwise they were too high
    else:
        print("Too high, try again.")
else:
    print("You're out of guesses. You lose!")
```

Sample Output

```
I'm thinking of a number between 1 and 100. Try to guess it.

Enter your guess: 10
Too low, try again.

Enter your guess: 20
Too low, try again.

Enter your guess: 30
Too low, try again.

Enter your guess: 40
Too low, try again.

Enter your guess: 50
Too low, try again.

Enter your guess: 60
Too low, try again.

Enter your guess: 70
Too low, try again.

Enter your guess: 80
Too low, try again.

Enter your guess: 90
Too high, try again.

Enter your guess: 81
Too low, try again.
You're out of guesses. You lose!
```

```
I'm thinking of a number between 1 and 100. Try to guess it.

Enter your guess: 50
Too high, try again.

Enter your guess: 25
Too low, try again.

Enter your guess: 37
Too high, try again.

Enter your guess: 31
Too high, try again.

Enter your guess: 28
Too high, try again.

Enter your guess: 26
You guessed it!
```

6.Input validation – Registration
Write a program which simulates registering a user account.

The user should be prompted to enter a valid username. A username is valid if it is at most 8 characters and contains no uppercase letters. Use a `while` loop to ensure that a valid username is entered.

The user should then be prompted to enter his/her password, and again to confirm it. Use a `while` loop to ensure that a) the two passwords match, and b) the password is at least 8 characters long.

Once a valid username and password has been entered, the message "Registration Complete" should be displayed.

Specification Table

Input	Processing	Output

username
password1
password2
	Input username
	While username not valid (<8 characters, or not lowercase)
	  Display "Username invalid"
	  Input username
	
	Input password1
	Input password2
	While password1 too short or password1 not equal to password2
	  Display "Passwords don't match"
	  Input password1
	  Input password2
	
	Display "Registration successful"
		username invalid
		passwords not match
		registration successful

Python Program

```
# Program Name: section02_solutions15_registration.py
# Purpose: To simulate registering a username and password
# Exampe of: input validation loop

print("Simulating user registration")

# input the username
username = input("Enter the username: ")

# keep repeating until it's valid
while len(username) > 8 or not username.islower():
    print("Invalid username, try again.")
    # input the username
    username = input("Enter the username: ")

# input the password and the password confirmation
password = input("Enter the password: ")
password2 = input("Confirm the password: ")

# keep repeating until the passwords match and are at least 8 characters long
while len(password) < 8 or password != password2:
    print("Invalid password. Try again.")
    # input the password and the password confirmation
    password = input("Enter the password: ")
    password2 = input("Confirm the password: ")

print()
print("Registration successful")
```

Sample Output

```
Simulating user registration

Enter the username: jbloggs

Enter the password: Secret1234

Confirm the password: Secret1234

Registration successful
```

```
Simulating user registration

Enter the username: JoeBloggs
Invalid username, try again.

Enter the username: JBloggs
Invalid username, try again.

Enter the username: jbloggs

Enter the password: Secret1234

Confirm the password: secret1234
Invalid password. Try again.

Enter the password: Secret1234

Confirm the password: Secret1234

Registration successful
```

7.Input Validation Loop – IP addresses

An IP address (version 4) takes the form `w.x.y.z` where `w x y` and `z` are numbers in the range 0 – 255, e.g. 192.168.34.10. It would be a mistake to any of the numbers was negative, or greater than 255.

Write a program which prompts the user to input a valid IP address. The program should use a counting loop to input the 4 numbers of the IP address, and use an input validation loop to ensure that the number is valid before proceeding. Once a valid number has been input, it should be added to the IP address.

Hints:

You'll need to input each number as an integer, and check if its between 0 and 255, inclusive.

If this is the first valid number (count is zero), then initialise the ip address string using:

```
ip_address = str(number)
```

where the `str` function converts the number to a string; then increase the count by 1.

For each subsequent valid number, add it to the ip address using:

```
ip_address = ip_address + "." + str(number)
```

or alternatively

```
ip_address += "." + str(number)
```

Specification Table

Input	Processing	Output
number password1 password2	For each number i from 0 to 3 (provided by range(4)) Input number While username not between 0 and 255 Display "Invalid number" Input number If i = 0 (first number) Set IP address to number (converted to a string using str) Else Add number of IP address, separated by a dot "." Display IP address	number invalid IP address

Python Program

```
# Program Name: section02_solutions16_validate_ipaddress.py
# Purpose: To validate the numbers in an IP Address (version 4)
# Exampe of: input validation loop

print("Program to validate the numbers in an IP Address (version 4)")

# input 4 valid numbers
for i in range(4):
    # input the number
    number = int(input("Enter the number: "))

    # keep repeating until the number is valid
    while not 0 <= number <= 255:
        number = int(input("Invalid number, try again: "))

    # number is valid. is it the first one?
    if i == 0:
        ip_address = str(number) # convert the number to a string
    else: # add it on to the existing ip address
        ip_address += "." + str(number)

print()
print(f"IP Address is: {ip_address}")
```

Sample Output

```
Program to validate the numbers in an IP Address (version 4)
Enter the number: 192
Enter the number: 168
Enter the number: 24
Enter the number: 10
IP Address is: 192.168.24.10

Program to validate the numbers in an IP Address (version 4)
Enter the number: 999
Invalid number, try again: 199
Enter the number: 256
Invalid number, try again: 257
Invalid number, try again: 255
Enter the number: 255
Enter the number: 255
IP Address is: 199.255.255.255
```

8.Weight Conversions

The conversion from Imperial to Metric weights, and vice versa, are:

1 pound = 0.453 592 37 kilogram

1 kilogram = 2.204 622 621 8 pound

Write a program which repeatedly displays a menu offering to convert from pounds to kilograms, and kilograms to pounds and then processes the users choice until s/he is finished. The program should display a suitable message if the user inputs an invalid choice.

Specification Table

Input	**Processing**	**Output**
choice pounds kilograms	Repeat for ever (use while True) Display a menu Input choice If choice = 0 break from loop Else if choice = 1 Input pounds Calculate kilograms Display kilograms Else if choice = 2 Input kilograms Calculate pounds Display pounds Else Dfisplay “Invalid choice” If i = 0 (first number) Set IP address to number (converted to a string using str) Else Add number of IP address, separated by a dot “.” Display IP address	menu kilograms pounds Invalid choice

Python Program

```
# Program Name: section02_solutions17_weight_conversions.py
# Purpose: To convert weights from Imperial to Metric and vice versa
# Exampe of: repeating menu

while True:
    print("Imperial and Metric Weight Conversions")
    print("1. Imperial to Metric")
    print("2. Metric to Imperial")
    print("0. Quit")
    choice = int(input("Enter your choice: "))

    if choice == 0:
        break
    elif choice == 1:
        pounds = float(input("Enter the weight in pounds: "))
        kg = pounds * 0.45359237
        print(f"Equivalent weight is: {kg:.1f}kg")
    elif choice == 2:
        kgs = float(input("Enter the weight in kilograms: "))
        pounds = kgs * 2.2046226218
        print(f"Equivalent weight is: {pounds:.1f}lbs")
    else:
        print("Invalid choice")

    print() # blank line
```

Sample Output

```
Imperial and Metric Weight Conversions
1. Imperial to Metric
2. Metric to Imperial
0. Quit

Enter your choice: 1

Enter the weight in pounds: 200
Equivalent weight is: 90.7kg

Imperial and Metric Weight Conversions
1. Imperial to Metric
2. Metric to Imperial
0. Quit

Enter your choice: 2

Enter the weight in kilograms: 80
Equivalent weight is: 176.4lbs

Imperial and Metric Weight Conversions
1. Imperial to Metric
2. Metric to Imperial
0. Quit

Enter your choice: 0
```

9.Validate Internet Domain
A domain name is an identification string that defines a realm of administrative autonomy, authority or control within the Internet. https://en.wikipedia.org/wiki/Domain_name Traditionally, a domain name can contain at most 253 characters, and can only consist of letters, numbers, dots . and dashes (hypens) -. Write a program which inputs and validates a domain name.

Specification Table

Input	Processing	Output
domain name	Input domain name For each character in the domain name If character is not alphanumeric and not a dot and not a dash Display "Invalid character" break from for loop Else Display "Valid Domain Name"	Invalid character Valid domain name

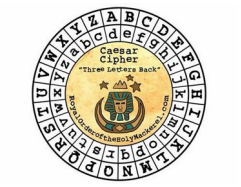
Python Program

```
# Program Name: section02_solutions18_validate_domain.py
# Purpose: To convert weights from Imperial to Metric and vice versa
# Exampe of: for loop to process a string

print("Program to validate an Internet Domain Name")

domain = input("Enter the domain name: ")

# check the length
if len(domain) > 253:
    print("Domain name exceeds 253 characters")
else:
    # Check the domain name for invalid characters
    for character in domain:
        if not character.isalnum() and character != '.' and character != '-':
            print(f"Invalid character: {character}")
            break
    else:
        print("Valid Domain name")
```

Sample Output

```
Enter the domain name: research.ait.ie
Valid Domain name

Enter the domain name: advanced_research.ait.ie
Invalid character: _
```

10.Caesar Cipher



Julius Caesar is credited with a basic substitution cipher in which he shifted each letter of the alphabet along by 3 places. The original message is referred to as the *plaintext*, represented in lowercase characters, and the enciphered message is referred to as the *ciphertext*, represented in uppercase characters.

a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z
D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C

So a is enciphered as D, b is enciphered as E, … The letters at the end of the alphabet (x, y and z) wrap-around to be substituted by the first three.

0	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**	**16**	**17**	**18**	**19**	**20**	**21**	**22**	**23**	**24**	**25**
a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z
D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C

Write a program which inputs a plaintext message, enciphers it using the Caesar Cipher, and then displays the ciphertext.

Specification Table

Input	**Processing**	**Output**

plaintext	Input the plaintext
	
	Convert the plaintext to lowercase
	
	For each character in the plaintext one at a time
	  If the character is a letter, convert it into a number representing its position in the alphabet
	  Add 3 to this number
	  Find the letter corresponding to this number, convert it to uppercase and add it to the ciphertext
	  Otherwise add the character on to the ciphertext
	
	Display the ciphertext
		ciphertext

Python Program

```
# Program Name: section02b_solutions10_caesar_cipher.py
# Purpose: To implement the Caesar Cipher
# Example Of: Control Structures - for loop
from string import ascii_lowercase, ascii_uppercase

print("Program to encipher a message using the Caesar Cipher")

# Input the message
plaintext = input("Enter the message: ")

# create an empty string representing the ciphertext
ciphertext =""

# For each character in the message
for character in plaintext.lower():
     # if it's a letter
     if character.islower():
          # convert it to a number
          number = ascii_lowercase.index(character)
          # add 3 to the number
          new_number = (number + 3) % 26
          # find the UPPERCASE letter for the new number
          new_letter = ascii_uppercase[new_number]
          # add the letter to the ciphertext
          ciphertext += new_letter
     # otherwise just add it on
     else:
           ciphertext += character

# print the ciphertext
print("The enciphered message is:", ciphertext)
```

Sample Output

```
Program to encipher a message using the Caesar Cipher

Enter the message: The Celts are attacking!
The enciphered message is: WKH FHOWV DUH DWWDFNLQJ!
```