**Computer Practical**

Python: Grade 12

*Rayhan Mohammad Riyaz*

*12 A*

**Contents**

|  |  |  |  |
| --- | --- | --- | --- |
| Chapter No. | Chapter Name | Page No. | Teacher’s Signature |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

# Chapter 1: Python Revision Tour

## Solved Problems (pg 36)

|  |  |
| --- | --- |
| 22 | **Question:** Write a program that asks the user to input number of seconds and then expresses it in terms of many minutes and seconds it contains. |
| **Code:**  total\_seconds = int(input("Number of Seconds: "))  minutes = total\_seconds//60  seconds = total\_seconds%60  print(total\_seconds, "is", minutes, "minutes and", seconds, "seconds.") |
| **Output:**  Number of Seconds: 393  393 is 6 minutes and 33 seconds. |
| 23 | **Question:** Write a program that repeatedly asks from users some numbers until string ‘done’ is typed. The program should print the sum of all numbers entered. |
| **Code:**  num\_sum = 0  while True:      num = input("Number to be added (or 'done' to stop): ")      if num=='done':          break      else:          num = int(num)      num\_sum += num  print("Sum of the numbers inputted:", num\_sum) |
| **Output:**  Number to be added (or 'done' to stop): 13  Number to be added (or 'done' to stop): 2  Number to be added (or 'done' to stop): -5  Number to be added (or 'done' to stop): done  Sum of the numbers inputted: 10 |
| 24 | **Question:** Write a program to print a square multiplication table as shown below |
| **Code:**  hor = input("Horizontal Width of Multiplication Table (default=10): ")  ver = input("Vertical Height of Multiplication Table (default=10): ")  if not hor:      hor=10  if not ver:      ver=10  max\_spaces = len(str(hor\*ver))  for i in range(1, ver+1):      for j in range(i, hor\*i+1, i):          print(j, end=' '\*(max\_spaces-len(str(j))))      print() |
| **Output:**  Horizontal Width of Multiplication Table (default=10):  Vertical Height of Multiplication Table (default=10):  1 2 3 4 5 6 7 8 9 10  2 4 6 8 10 12 14 16 18 20  3 6 9 12 15 18 21 24 27 30  4 8 12 16 20 24 28 32 36 40  5 10 15 20 25 30 35 40 45 50  6 12 18 24 30 36 42 48 54 60  7 14 21 28 35 42 49 56 63 70  8 16 24 32 40 48 56 64 72 80  9 18 27 36 45 54 63 72 81 90  10 20 30 40 50 60 70 80 90 100 |

## Type C

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 1 | **Question:** Write a program to print one of the words negative, zero or positive, according to whether variable x is less than zero, zero or greater than zero, respectively. | | | |
| **Code:**  x = int(input("Number: "))  print(x, "is", end=" ")  if x<0:      print("negative (less than 0).")  elif x==0:      print("zero (0).")  else:      print("positive (greater than 0).") | | | |
| **Output:** | | | |
| Number: 7  7 is positive (greater than 0). | Number: 0  0 is zero (0). | | Number: -3  -3 is negative (less than 0). |
| 3 | **Question:** Write a Python program that calculates and prints the number of seconds in a year | | | |
| **Code:**  num\_years = 1  num\_days = num\_years\*365  num\_hours = num\_days\*24  num\_minutes = num\_hours\*60  num\_seconds = num\_minutes\*60  print("Number of seconds in", num\_years, "years is", num\_seconds) | | | |
| **Output:**  Number of seconds in 1 years is 31536000 | | | |
| 5 | **Question:** Write a program that asks the user the day number in a year in the range 2 to 365 and asks the first day of the year – Sunday, Monday, Tuesday, etc. Then the program should display the day on the day-number that has been inputted. | | | |
| **Code:**  first\_day\_str = input("First day of the year (in lowercase): ")  date = int(input("Which day number of the year do you want to know the day of (2 to 365): "))  first\_day=0  if first\_day\_str=="monday":  first\_day=1  elif first\_day\_str=="tuesday":  first\_day=2  elif first\_day\_str=="wednesday":  first\_day=3  elif first\_day\_str=="thursday":  first\_day=4  elif first\_day\_str=="friday":  first\_day=5  elif first\_day\_str=="saturday":  first\_day=6  day\_num = (first\_day + date - 1) % 7  day = "Sunday"  if day\_num == 1:  day = "Monday"  elif day\_num == 2:  day = "Tuesday"  elif day\_num == 3:  day = "Wednesday"  elif day\_num == 4:  day = "Thursday"  elif day\_num == 5:  day = "Friday"  elif day\_num == 6:  day = "Saturday"  print(date, "day of the year with the first day", first\_day\_str, "is", day) | | | |
| **Output:**  First day of the year (in lowercase): tuesday  Which day number of the year do you want to know the day of (2 to 365): 19  19 day of the year with the first day tuesday is Saturday | | | |
| 7 | **Question:** Write a program that reads and integer N from the keyboard comptu8es and displays the sum of the numbers from N to 2\*N if N is non negative. If N is a negative number, then it’s the sum of the numbers from 2\*N to N. The starting and ending points are included in the sum. | | | |
| **Code:**  N = int(input("integer N: "))  sum=0  if N>=0:      for i in range(N, 2\*N+1):          sum+=i  else:      for i in range(2\*N, N+1, ):          sum+=i  print(sum) | | | |
| **Output:** | | | |
| integer N: -4  -30 | | integer N: 8  108 | |

# Chapter 2: Python Revision Tour-II

## Programs (pg 71)

|  |  |
| --- | --- |
| 2.4 | **Question:** Give three lists as list1 = [‘a’, ‘b’, ‘c’], list2 = [‘h’, ‘i’,‘t’] and list3 = [‘0’, ‘1’, ‘2’]. Write a program that adds individual elements of list2 and list3 to list1. The resultant list should be in the order of list3, list1 and then list2. |
| **Code:**  list1 = ['a', 'b', 'c']  list2 = ['h', 'i', 't']  list3 = ['0', '1', '2']  print("List 1:", list1)  print("List 2:", list2)  print("List 3:", list3)  for i in list3[::-1]:  list1.insert(0, i)  for i in list2:  list1.append(i)  print("Resultant List:", list1) |
| **Output:**  List 1: ['a', 'b', 'c']  List 2: ['h', 'i', 't']  List 3: ['0', '1', '2']  Resultant List: ['0', '1', '2', 'a', 'b', 'c', 'h', 'i', 't'] |
| 2.5 | **Question:** Write a program that finds an element’s index/position in a tuple without using index() |
| **Code:**  lst = eval(input("List: "))  element = eval(input("Element: "))  indices = []  for i in range(len(lst)):  if lst[i]==element:  indices.append(i)  print("Element's posible index/position: ", end="")  for i in indices:  print(i, end=" ") |
| **Output:**  List: [1,2,3,'a', 1]  Element: 1  Element's posible index/position: 0 4 |

## Type C

|  |  |
| --- | --- |
| 2 | **Question:** Write a program that should prompt the user to type some sentence(s) followed by “enter”. It should then print the original sentence(s) and the following statistics relating to the sentence(s): Number of words, Number of characters (including white-spaces and punctuation) and Percentage of characters that are alpha numeric. |
| **Code:**  sentences = input("Type a few sentence(s) (and press enter): ")  print("Original Sentence(s):")  print(sentences)  word\_count = len(sentences.split(" "))  char\_count = len(sentences)  alnum\_count = 0  for i in sentences:  if i.isalnum():  alnum\_count+=1  alnum\_percent = alnum\_count/char\_count \* 100  print("Number of words:", word\_count)  print("Number of characters:", char\_count)  print("Percentage of alpha-numeric characters: ",\ alnum\_percent, "%", sep="") |
| **Output:**  Type a few sentence(s) (and press enter): Large programs of size 100 MB to 200 MB are generally hard to store; except in case of integers. As, they have numeric values.  Original Sentence(s):  Large programs of size 100 MB to 200 MB are generally hard to store; except in case of integers. As, they have numeric values.  Number of words: 24  Number of characters: 126  Percentage of alpha-numeric characters: 78.57142857142857% |
| 3 | **Question:** Write a program that takes any two lists L and M of the same size and adds their elements together to form a new list N whose elements are sums of the corresponding elements in L and M. For instance, if L= [3, 1, 4] and M= [1, 5, 9], then N should equal [4, 6, 13]. |
| **Code:**  L = eval(input("List L: "))  M = eval(input("List M: "))  N = list()  for i in range(len(L)):  N.append(L[i]+M[i])  print("List N:", N) |
| **Output:**  List L: [3,1,4]  List M: [1,5,9]  List N: [4, 6, 13] |
| 8 | **Question:** Write a Python program that creates a tuple storing first 9 terms of Fibonacci series. |
| **Code:**  n=9  fib\_tup = (0,1)  for i in range(n-len(fib\_tup)):  fib\_tup+=sum(fib\_tup[-2:]),  print("Tuple with Fibonacci Series' (",n,"terms):", fib\_tup) |
| **Output:**  Tuple with Fibonacci Series' ( 9 terms): (0, 1, 1, 2, 3, 5, 8, 13, 21) |
| 9 | **Question:** Create a dictionary whose keys are month names and whose values are the number of days in the corresponding months.   1. Ask the user to enter a month name and use the dictionary to tell them how many days are in the month 2. Print out all the keys in alphabetical order 3. Print out all the months with 31 days 4. Print out the key-value pairs sorted by the number of days in each month |
| **Code:**  month\_days = {'January': 31, 'February': 28, 'March': 31, 'April': 30,  'May': 31, 'June': 30, 'July': 31, 'August': 31,  'September': 30, 'October': 31, 'November': 30, 'December': 31}  month\_name = input("Month Name:").strip().capitalize()  print("Number of days in", month\_name, "=", month\_days[month\_name])  keys = sorted(month\_days.keys())  print("All keys (months) in alphabetical order:")  print(", ".join(keys))  month\_31\_days = []  for k, v in month\_days.items():  if v==31:  month\_31\_days.append(k)  print("All months with 31 days:")  print(", ".join(month\_31\_days))  pairs = list(month\_days.items())  for i in range(1,len(pairs)):  if pairs[i][1]<pairs[i-1][1]:  for j in range(i):  if pairs[j][1]>pairs[i][1]:  pairs.insert(j, pairs[i])  pairs.pop(i+1)  print("All key-value (month-day) pairs sorted by number of days in each month:")  print(pairs) |
| **Output:**  Month Name:JANuary  Number of days in January = 31  All keys (months) in alphabetical order:  April, August, December, February, January, July, June, March, May, November, October, September  All months with 31 days:  January, March, May, July, August, October, December  All key-value (month-day) pairs sorted by number of days in each month:  [('February', 28), ('April', 30), ('June', 30), ('September', 30), ('November', 30), ('January', 31), ('March', 31), ('May', 31), ('July', 31), ('August', 31), ('October', 31), ('December', 31)] |

# Chapter: Working with Functions

|  |  |
| --- | --- |
| 1 | **Question:** Write a function that takes amount in dollars and dollar to rupee conversion price; it then returns the amount converted to rupees. Create the function in both void and non-void forms. |
| **Code:**  def usd\_to\_inr(usd):      inr = usd\*83.48 #As per 10 May 2024      return inr  def void\_usd\_to\_inr():      usd = float(input("US Dollars: "))      inr = usd\*83.48 #As per 10 May 2024      print("Indian Rupees:", inr)  usd = float(input("US Dollars: "))  inr = usd\_to\_inr(usd)  print("Indian Rupees:", inr)  void\_usd\_to\_inr() |
| **Output:**  US Dollars: 5  Indian Rupees: 417.40000000000003  US Dollars: 5  Indian Rupees: 417.40000000000003 |
| 2 | **Question:** Write a function to calculate volume of a box with appropriate default values for its parameters. Your function should have the following input parameters: length, width and height of box. |
| **Code:**  def volume\_box(length=1, width=1, height=1):      volume = length\*width\*height      return volume  print(volume\_box())  print(volume\_box(2,3,3))  print(volume\_box(15, 2))  print(volume\_box(3, width=2)) |
| **Output:**  Volume of Box (l=def, w=def, h=def): 1  Volume of Box (l=2, w=3, h=3): 18  Volume of Box (l=15, w=2, h=def): 30  Volume of Box (l=3, w=2, h=def): 6 |
| 3 | **Question:** Write a program to have following functions:   1. A function that takes a number as argument and calculates cube for it. The function does not return a value, if there is no value passed to the function in function call, the function should calculate cube of 2. 2. A function that takes two char arguments and return True if both the arguments are equal, otherwise False |
| **Code:**  def cube (n=2):      n\_cubed = n\*\*3      print("Cube of", n, "is", n\_cubed)  cube()  cube(5)  def arg\_equals(arg\_1, arg\_2):      return arg\_1==arg\_2  arg\_1 = input("Char Argument 1: ")  arg\_2 = input("Char Argument 2: ")  print(arg\_equals(arg\_1, arg\_2)) |
| **Output:**  Cube of 2 is 8  Cube of 5 is 125  Char Argument 1: a  Char Argument 2: a  True |
| 4 | **Question:** Write a function that receives two numbers and generates a random number from the range. Using this function, the main program should be able to print three numbers randomly. |
| **Code:**  import random  def random\_range(num\_1, num\_2):  return random.randrange(num\_1, num\_2)  num\_1 = int(input("Number 1: "))  num\_2 = int(input("Number 2: "))  for i in range(3):  print("Random number", i+1, ":", random\_range(num\_1, num\_2)) |
| **Output:**  Number 1: 2  Number 2: 13  Random number 1 : 11  Random number 2 : 4  Random number 3 : 2 |
| 5 | **Question:** Write a function that receives two string arguments and checks whether they are same-length strings |
| **Code:**  def same\_len(str\_1, str\_2):  return len(str\_1)==len(str\_2)  string\_1 = input("String 1: ")  string\_2 = input("String 2: ")  print(same\_len(string\_1, string\_2)) |
| **Output:**  String 1: apple  String 2: lemon  True |
| 8 | **Question:** Write a function that takes two numbers and returns the number that as minimum one’s digits. |
| **Code:**  def return\_min\_one\_digit(num\_1, num\_2):  return [num\_1, num\_2][num\_1%10>num\_2%10]  number\_1 = int(input("Number 1: "))  number\_2 = int(input("Number 2: "))  print("Minimum One's digit: ", return\_min\_one\_digit(number\_1, number\_2)) |
| **Output:**  Number 1: 471  Number 2: 278  Minimum One's digit: 471 |
| 9 | **Question:** Write a program that generates a series using a function which takes first and last values of the series and then generates four terms that are equidistant |
| **Code:**  def series\_list\_generator(start, end, n=4):  interval = (end-start)/(n-1)  if int(interval)==interval:  interval = int(interval)  series = [start, end]  for i in range(n-2):  series.insert(-1, series[-2]+interval)  return series  num\_start = int(input("First Value of Series: "))  num\_last = int(input("Last Value of Series: "))  print(\*series\_list\_generator(num\_start, num\_last)) |
| **Output:**  First Value of Series: 1  Last Value of Series: 7  1 3 5 7 |
| Q | **Question:** Write a menu-based program to check base of input number and then convert it to binary, octal, decimal and hexadecimal. |
| **Code:**  def binary(n):  return bin(n)[2:]  def octal(n):  return oct(n)[2:]  def decimal(n):  return str(n)  def hexadecimal(n):  return hex(n)[2:]  options = {'b': [binary, 'Binary', 2],  'o': [octal, 'Octal', 8],  'd': [decimal, 'Decimal', 10],  'h': [hexadecimal, 'Hexadecimal', 16]}  def menu():  print()  for i in options:  print(i, ") To convert from ", options[i][1], sep='')  print("Type exit to quit.")  print("Type letter to corresponding input base")  option = input("Option: ").strip()[0]  if option not in options:  return "exit"  number = input(options[option][1]+" Number: ")  convert(option, number)  def convert(option, number):  print("\n", number, "in base", options[option][2], "is:")  number = int(str(number), options[option][2])  output\_dict = dict()  for j in options:  if j!=option:  output\_dict[options[j][1]] = options[j][0](number)  for output in output\_dict:  print(output\_dict[output], "in", output)  while True:  if menu()=="exit":  break |
| **Output:**  b) To convert from Binary  o) To convert from Octal  d) To convert from Decimal  h) To convert from Hexadecimal  Type exit to quit.  Type letter to corresponding input base  Option: b  Binary Number: 10101  10101 in base 2 is:  25 in Octal  21 in Decimal  15 in Hexadecimal  b) To convert from Binary  o) To convert from Octal  d) To convert from Decimal  h) To convert from Hexadecimal  Type exit to quit.  Type letter to corresponding input base  Option: o  Octal Number: 17  17 in base 8 is:  1111 in Binary  15 in Decimal  f in Hexadecimal  b) To convert from Binary  o) To convert from Octal  d) To convert from Decimal  h) To convert from Hexadecimal  Type exit to quit.  Type letter to corresponding input base  Option: d  Decimal Number: 10  10 in base 10 is:  1010 in Binary  12 in Octal  a in Hexadecimal  b) To convert from Binary  o) To convert from Octal  d) To convert from Decimal  h) To convert from Hexadecimal  Type exit to quit.  Type letter to corresponding input base  Option: h  Hexadecimal Number: af  af in base 16 is:  10101111 in Binary  257 in Octal  175 in Decimal  b) To convert from Binary  o) To convert from Octal  d) To convert from Decimal  h) To convert from Hexadecimal  Type exit to quit.  Type letter to corresponding input base  Option: exit |

# Chapter 4: Data Structures II- Stacks and Queues using Lists

## NCERT

|  |  |
| --- | --- |
| 7 | **Question:** Write a program to create a Stack for storing only odd numbers out of all the numbers entered by the user. Display the content of the Stack along with the largest odd number in the Stack. |
| **Code:**  def push(stack, item):  stack.append(item)  def pop(stack):  if stack:  return stack.pop()  def odd\_push(num):  if num%2:  push(stack, num)  def get\_largest(stack):  elem = pop(stack)  large = elem  while elem!=None:  if elem > large:  large = elem  elem = pop(stack)  return large  n = int(input("Count: "))  stack = []  for i in range(n):  number = int(input("Number: "))  odd\_push(number)    print("Stack: ", stack)  large = get\_largest(stack)  print("Largest in Stack: ", large) |
| **Output:**  Count: 5  Number: 22  Number: 11  Number: 7  Number: 33  Number: 5  Stack: [11, 7, 33, 5]  Largest in Stack: 33 |

## Type C

|  |  |  |
| --- | --- | --- |
| 2 | **Question:** A line of text is read from the input terminal into a stack. Write a program to output the string in the reverse order, each character appearing twice. (c d e->ee dd cc) | |
| **Code:**  def push(stack, item, repeat=1):  while repeat>0:  stack.append(item)  repeat-=1  def pop(stack):  if stack:  return stack.pop()  inp\_text = input("Input Text Line: ")  stack=[]  for i in inp\_text:  push(stack, i, 2)  out\_text = ""  while stack:  out\_text+=pop(stack)  print("Output Text Line: ", out\_text) | |
| **Output:**  Input Text Line: a b c d e  Output Text Line: ee dd cc bb aa | |
| 3 | **Question:** Write a function in Python POP (Arr), where Arr is a stack implemented by a list of numbers. The function returns the value deleted from the stack. | |
| **Code:**  def POP(Arr):  if Arr:  return Arr.pop()  Arr = eval(input("Input Stack (in list form): "))  while True:  inp = input("Type pop to POP from Stack (quit to exit): ")  if inp == "quit":  break  else:  print("Popped element:", POP(Arr))  print("New Stack:", Arr) | |
| **Output:**  Input Stack (in list form): [1,2,3,4]  Type pop to POP from Stack (quit to exit): pop  Popped element: 4  New Stack: [1, 2, 3]  Type pop to POP from Stack (quit to exit): pop  Popped element: 3  New Stack: [1, 2]  Type pop to POP from Stack (quit to exit): quit | |
| 4 | **Question:** Write a program that depending upon the user’s choice, either pushes or pops and element in a stack. The elements are shifted towards right so that top always remains at 0th index. | |
| **Code:**  def push(stack, item):  stack.insert(0, item)  def pop(stack):  if stack:  return stack.pop(0)  stack = []  while True:  print("Commands:", "pop: POP the top element (1st)", "push {element}: PUSH the {element} above top (1st)", "display: DISPLAY the stack", "(anything else): Exit the Loop", sep='\n')  inp = input("Command:")  if inp=="pop":  pop(stack)  elif inp.startswith("push"):  push(stack, inp.partition(' ')[-1])  elif inp=="display":  print(stack)  else:  break  print() | |
| **Output:**  Commands:  pop: POP the top element (1st)  push {element}: PUSH the {element} above top (1st)  display: DISPLAY the stack  (anything else): Exit the Loop  Command:push 5  Commands:  pop: POP the top element (1st)  push {element}: PUSH the {element} above top (1st)  display: DISPLAY the stack  (anything else): Exit the Loop  Command:push 3  Commands:  pop: POP the top element (1st)  push {element}: PUSH the {element} above top (1st)  display: DISPLAY the stack  (anything else): Exit the Loop  Command:pop  Commands:  pop: POP the top element (1st)  push {element}: PUSH the {element} above top (1st)  display: DISPLAY the stack  (anything else): Exit the Loop  Command:display  ['5']  Commands:  pop: POP the top element (1st)  push {element}: PUSH the {element} above top (1st)  display: DISPLAY the stack  (anything else): Exit the Loop  Command:exit | |
| 8 | **Question:** Each node of a STACK contains the following information: (i) Pin code and (ii) Name of City. Write a program to implement following operations in above stack:   1. PUSH() to push a node into the stack 2. POP() to remove a node from the stack | |
| **Code:**  def PUSH(stack, node):  stack.append(node)  def POP(stack):  if stack:  return stack.pop()    stack = []  while True:  print("Commands:", "PUSH: add node", "POP: remove node", "EXIT: exit", sep="\n")  inp = input("Command: ")  if inp=="PUSH":  pin = input("Pin Code of Node City: ")  name = input("Name of Node City: ")  PUSH(stack, {"pincode":pin, "name":name})  elif inp=="POP":  POP(stack)  elif inp=="EXIT":  break  print(stack) | |
| **Output:**  Commands:  PUSH: add node  POP: remove node  EXIT: exit  Command: PUSH  Pin Code of Node City: 123456  Name of Node City: City#ABC  [{'pincode': '123456', 'name': 'City#ABC'}]  Commands:  PUSH: add node  POP: remove node  EXIT: exit  Command: POP  []  Commands:  PUSH: add node  POP: remove node  EXIT: exit  Command: EXIT | |
| 10 | **Question:** Write PUSH (Books) and POP (Books) methods, in python to add Books and remove Books considering them to act as Push and Pop operations of Stack | |
| **Code:**  def PUSH(books, book\_name):  books.append(book\_name)  def POP(books):  if books:  return books.pop()  books = []  while True:  print("Commands:", "PUSH {BOOK\_NAME}: add {BOOK\_NAME} to books", "POP: remove book from books", "EXIT: exit", sep="\n")  inp = input("Command: ")  if inp.startswith("PUSH"):  PUSH(books, inp.partition(' ')[-1])  elif inp=="POP":  POP(books)  elif inp=="EXIT":  break  print(books) | |
| **Output:**  Commands:  PUSH {BOOK\_NAME}: add {BOOK\_NAME} to books  POP: remove book from books  EXIT: exit  Command: PUSH Harry Potter  ['Harry Potter']  Commands:  PUSH {BOOK\_NAME}: add {BOOK\_NAME} to books  POP: remove book from books  EXIT: exit  Command: PUSH Physics NCERT 1  ['Harry Potter', 'Physics NCERT 1']  Commands:  PUSH {BOOK\_NAME}: add {BOOK\_NAME} to books  POP: remove book from books  EXIT: exit  Command: POP  ['Harry Potter']  Commands:  PUSH {BOOK\_NAME}: add {BOOK\_NAME} to books  POP: remove book from books  EXIT: exit  Command: EXIT | |
| 13 | **Question:** Write a function in Python PUSH (Arr), where Arr is a list of numbers. From this list, push all numbers divisible by 5 into a stack implemented by using a list. Display the stack if it has at least one element otherwise display appropriate error message. | |
| **Code:**  def push(stack, element, check\_divisible=1):  if element%check\_divisible==0:  stack.append(element)  def PUSH(Arr):  num\_stack = []  for num in Arr:  push(num\_stack, num, 5)  return num\_stack  num\_list = eval(input("List of Numbers: "))  num\_stack = PUSH(num\_list)  if num\_stack:  print("Stack with elements divisible by 5:", num\_stack)  else:  print("ERROR: Empty Stack Generated") | |
| **Output:** | |
| List of Numbers: [1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20]  Stack with elements divisible by 5: [5, 10, 15, 20] | List of Numbers: [1,4,9]  ERROR: Empty Stack Generated |

# Chapter: Exception Handling

|  |  |
| --- | --- |
| 1 | **Question:** Write a program that prompts the user to input an integer and raises a ValueError if input is not a valid integer. |
| **Code:**  while True:  try:  inp = input("Input Integer: ")  if not ((inp[0]=='-' and inp[1:].isdigit()) or inp.isdigit()):  raise ValueError  except:  if inp=="exit":  break  print("ValueError: Non-integer convertible type")  else:  integer = int(inp)  print("Integer:", integer) |
| **Output:**  Input Integer: -4  Integer: -4  Input Integer: .5  ValueError: Non-integer convertible type  Input Integer: a  ValueError: Non-integer convertible type  Input Integer: exit |
| 2 | **Question:** Write a program that reads two numbers and raises TypeError if input is not numerical. |
| **Code:**  while True:  try:  num1 = num2 = None  num1 = input("Number 1: ")  num2 = input("Number 2: ")  if not (num1.isnumeric() and num2.isnumeric()):  raise TypeError  except TypeError:  print("TypeError: Inputted Number is not of numeric data-type")  else:  sum = float(num1) + float(num2)  print("Sum:", sum)  finally:  if\_exit = input("Exit? ")  if if\_exit=="yes":  break |
| **Output:**  Number 1: 1  Number 2: 3  Sum: 4.0  Exit? no  Number 1: 5  Number 2: a  TypeError: Inputted Number is not of numeric data-type  Exit? yes |
| 3 | **Question:** Write a program that reads a list and raises IndexError if for any operation index is out of range. |
| **Code:**  lst = eval(input("Numeric List: "))  while True:  try:  print("Options:", "ADD {NUMBER} {INDEX}", "RMV {INDEX}", "EXIT", sep="\n")  cmd = input("Command: ").split(' ')  if cmd[0] == "EXIT":  break  else:  cmd[1], cmd[-1] = int(cmd[1]), int(cmd[-1])  if cmd[0]=="ADD":  if cmd[-1] > len(lst):  raise IndexError  else:  lst.insert(cmd[-1], cmd[1])  elif cmd[0]=="RMV":  if cmd[-1] > len(lst):  raise IndexError  else:  lst.pop(cmd[-1])    except IndexError:  print("IndexError: List index inputted is out of range")  finally:  print("List:", lst) |
| **Output:**  Numeric List: [1,3,4]  Options:  ADD {NUMBER} {INDEX}  RMV {INDEX}  EXIT  Command: ADD 5 4  IndexError: List index inputted is out of range  List: [1, 3, 4]  Options:  ADD {NUMBER} {INDEX}  RMV {INDEX}  EXIT  Command: RMV 3  IndexError: List index inputted is out of range  List: [1, 3, 4]  Options:  ADD {NUMBER} {INDEX}  RMV {INDEX}  EXIT  Command: ADD 5 1  List: [1, 5, 3, 4]  Options:  ADD {NUMBER} {INDEX}  RMV {INDEX}  EXIT  Command: EXIT  List: [1, 5, 3, 4] |
| 4 | **Question:** Write a program that read details (name, subject, marks) of students for result preparation. Incorporate all exception handling codes such as ValueError, IndexError, ZeroDivisionError, User defined Exceptions. |
| **Code:**  std\_db = dict()  percentage\_db = dict()  error\_types = {'EXISTING\_KEY': False, 'SUBJECT\_COUNT\_VALUE': False, 'OPTION\_ERROR': False, 'SUBJECT\_MARK\_VALUE' : False, 'INDEX\_VALUE': False, 'MARK\_BEYOND\_RANGE': False}  menu = "\n".join(["Options:", "1) ADD STUDENT INFO", "2) EDIT STUDENT INFO", "3) REMOVE STUDENT INFO", "4) CALCULATE STUDENT", "5) DISPLAY MENU", "6) EXIT"])  print(menu)  while True:      for error in error\_types:          error\_types[error] = False      try:          print()          option = input("Option:")          if option == '1':              name = input("Name: ")              if name in std\_db:                  error\_types['EXISTING\_KEY'] = True                  raise Exception("Key already exists in database")              subject\_count = input("Subject Count: ")              if not subject\_count.isdigit():                  error\_types['SUBJECT\_COUNT\_VALUE'] = True                  raise ValueError              mark\_list = []              for i in range(int(subject\_count)):                  subject\_name = input(str(i+1) + " Subject Name: ")                  subject\_mark = input(str(i+1) + " Subject Mark: ")                  if not subject\_mark.isnumeric():                      error\_types['SUBJECT\_MARK\_VALUE'] = True                      raise ValueError                  if float(subject\_mark)>100:                      error\_types['MARK\_BEYOND\_RANGE'] = True                      raise Exception("Mark beyond range")                  mark\_list.append([subject\_name, float(subject\_mark)])              std\_db[name] = mark\_list          elif option=='2':              name = input("Name: ")              if name not in std\_db:                  raise KeyError              print(std\_db[name], *end*=" ")              index = input("Which Index to edit? ")              if not index.isdigit():                  error\_types['INDEX\_VALUE'] = True                  raise ValueError              index = int(index)              if index > len(std\_db[name]):                  raise IndexError              subject\_name = input("Subject Name: ")              subject\_mark = input("Subject Mark: ")              if not subject\_mark.isnumeric():                  error\_types['SUBJECT\_MARK\_VALUE'] = True                  raise ValueError              if float(subject\_mark)>100:                  error\_types['MARK\_BEYOND\_RANGE'] = True                  raise Exception("Mark beyond range")              std\_db[name][index] = [subject\_name, float(subject\_mark)]          elif option=='3':              name = input("Name: ")              if name not in std\_db:                  raise KeyError              del std\_db[name]          elif option=='4':              name = input("Name: ")              if name not in std\_db:                  raise KeyError              if len(std\_db[name]) == 0:                  raise ZeroDivisionError              total = 0              for mark in std\_db[name]:                  total += mark[1]              percentage = total/len(std\_db[name])              percentage\_db[name] = percentage          elif option=='5':              print(menu)          elif option=='6':              print("Exitted")              break          else:              error\_types['OPTION\_ERROR'] = True              raise Exception('OptionError')      except ValueError:          ERROR = None          for error in error\_types:              if error\_types[error]:                  ERROR = error                  break          print("ValueError (", ERROR, ") : Inputted Data is not of the appropriate datatype for value conversion", *sep*="")      except KeyError:          print("KeyError: Key not present in database")      except IndexError:          print("IndexError: Inputted index is out of range")      except ZeroDivisionError:          print("ZeroDivision: Subject Count is 0. Percentage incalculable")      except Exception:          ERROR = None          for error in error\_types:              if error\_types[error]==True:                  ERROR = error                  break          print("Exception (", ERROR, ") : Application Error", *sep*="")      else:          print("Student Database:", std\_db)          print("Percentage Database:", percentage\_db) |
| **Output:**  Options:  1) ADD STUDENT INFO  2) EDIT STUDENT INFO  3) REMOVE STUDENT INFO  4) CALCULATE STUDENT  5) DISPLAY MENU  6) EXIT  Option:1  Name: James  Subject Count: a  ValueError (SUBJECT\_COUNT\_VALUE) : Inputted Data is not of the appropriate datatype for value conversion  Option:1  Name: James  Subject Count: 1  1 Subject Name: Math  1 Subject Mark: a  ValueError (SUBJECT\_MARK\_VALUE) : Inputted Data is not of the appropriate datatype for value conversion  Option:1  Name: James  Subject Count: 2  1 Subject Name: Math  1 Subject Mark: 98  2 Subject Name: Science  2 Subject Mark: 89  Student Database: {'James': [['Math', 98.0], ['Science', 89.0]]}  Percentage Database: {}  Option:19  Exception (OPTION\_ERROR) : Application Error  Option:a  Exception (OPTION\_ERROR) : Application Error  Option:1  Name: James  Exception (EXISTING\_KEY) : Application Error  Option:2  Name: Mark  KeyError: Key not present in database  Option:2  Name: James  [['Math', 98.0], ['Science', 89.0]] Which Index to edit? 3  IndexError: Inputted index is out of range  Option:2  Name: James  [['Math', 98.0], ['Science', 89.0]] Which Index to edit? 0  Subject Name: Mathematics  Subject Mark: 101  Exception (MARK\_BEYOND\_RANGE) : Application Error  Option:2  Name: James  [['Math', 98.0], ['Science', 89.0]] Which Index to edit? 0  Subject Name: Mathematics  Subject Mark: 99  Student Database: {'James': [['Mathematics', 99.0], ['Science', 89.0]]}  Percentage Database: {}  Option:5  Options:  1) ADD STUDENT INFO  2) EDIT STUDENT INFO  3) REMOVE STUDENT INFO  4) CALCULATE STUDENT  5) DISPLAY MENU  6) EXIT  Student Database: {'James': [['Mathematics', 99.0], ['Science', 89.0]]}  Percentage Database: {}  Option:4  Name: James  Student Database: {'James': [['Mathematics', 99.0], ['Science', 89.0]]}  Percentage Database: {'James': 94.0}  Option:3  Name: James  Student Database: {}  Percentage Database: {'James': 94.0}  Option:6  Exitted |

|  |  |
| --- | --- |
|  | Options:  1) ADD STUDENT INFO  2) EDIT STUDENT INFO  3) REMOVE STUDENT INFO  4) CALCULATE STUDENT  5) DISPLAY MENU  6) EXIT  Option:1  Name: James  Subject Count: 0  Student Database: {'James': []}  Percentage Database: {}  Option:4  Name: James  ZeroDivision: Subject Count is 0. Percentage incalculable  Option:6  Exitted |

# Chapter: File Handling

|  |  |
| --- | --- |
| 1 | **Question:** Question |
| **Code:**  Code |
| **Output:**  Output |