|  |  |  |  |
| --- | --- | --- | --- |
| **Charotar University of Science and Technology Chandubhai S Patel Institute of Technology**  **U & P U. Patel Department of Computer Engineering**  **Lesson Planning (Practical)** | | | |
| **Academic Year: December-April, 2018-19** | | **Semester: 6th** | |
| **Subject Coordinator: Divyesh Patel** | | | |
| **Subject Teacher Name: Divyesh Patel, Hardik Mandora** | | | |
| **Subject:** | **CE376: PROGRAMMING IN PYTHON (Elective-II)** | **Hrs./Week: 2** | |
| **Practical No** | **Practical** | **Extra** | **Hours** |
| 1 | Installation of Configuration of Python. Along with its all major editors. |  | 2 |
| 2 | Create a program that asks the user to enter their name and their age. Print out a message addressed to them that tells them the year that they will turn 100 years old.  **from** datetime **import** datetime **class** user:  **def** \_\_init\_\_(self,name,age):  self.name=name  self.age=age  user1=user(input(**"enter your name:"**),input(**"enter your age:"**)) x=datetime.now() y=x.year-int(user1.age)+100  print(**"{} will turn 100 in year {}"**.format(user1.name,y)) | 1. Add on to the previous program by asking the user for another number and printing out that many copies of the previous message. (Hint: order of operations exists in Python)   y=input(**"enter a message"**) x=input(**"enter a number of times you have to print "**)  print(y\*int(x))   1. Print out that many copies of the previous message on separate lines. (Hint: the   string "\n is the same as pressing the ENTER button)  y=str(input(**"enter a message"**)) x=input(**"enter a number of times you have to print "**) z=y+**"\n"** print(z\*int(x)) | 2 |
| Ask the user for a number. Depending on whether the number is even or odd, print out an appropriate message to the user. Hint: how does an even / odd number react differently when divided by 2?  x=input(**"enter a number"**) **if** int(x)%2==0:  print(**"the number is even"**) **else**:  print(**"number is odd"**) | 1. If the number is a multiple of 4, print out a different message.   n=input(**"enter a number"**) **if**(int(n)%4==0):  print(**"it is multiple of 4"**)   1. Ask the user for two numbers: one number to check (call it num) and one number to divide by (check). If check divides evenly into num, tell that to the user. If not, print a different appropriate message.   x=int(input(**"enter a check"**)) y=int(input(**"enter a number to divide by check"**))  **if**(y%x==0):  print(**"check divides your number evenly"**) |
| Take a list, say for example this one: a = [1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89]  and write a program that prints out all the elements of the list that are less  than 5.  a=[1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89]  **for** i **in** a:  **if**(i<5):  print(i) | 1. Instead of printing the elements one by one, make a new list that has all the elements less than 5 from this list in it and print out this new list.   a=[1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89] k=[] **for** i **in** a:  **if**(i<5):  k.append(i)  print(k)   1. Write this in one line of Python.   a=[1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89] k=[i **for** i **in** a **if**(i<5)]  print(k)   1. Ask the user for a number and return a list that contains only elements from the original list a that are smaller than that number given by the user.   k=int(input(**"enter a number"**)) print([i **for** i **in** range(0,k)]) |
| 3 | Create a program that asks the user for a number and then prints out a list of all the divisors of that number. (If you don’t know what a divisor is, it is a number that divides evenly into another number. For example, 13 is a  divisor of 26 because 26 / 13 has no remainder.)  x=int(input(**"enter a number:"**)) **for** i **in** range(1,int(x/2)+1):  **if**(x%i==0):  print(i) print(x) |  | 2 |
| Take two lists, say for example these two:  a = [1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89]  b = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13]  and write a program that returns a list that contains only the elements that are common between the lists (without duplicates). Make sure your program works on two lists of different sizes.  a = [1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89] b = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13]  common=set()  **for** i **in** a:  **for** j **in** b:  **if**(i==j):  common.add(i)     print(common) | 1. Randomly generate two lists to test this   **import** random a = [random.randrange(0,100) **for** i **in** range(0,random.randrange(0,100))]  b = [random.randrange(0,100) **for** i **in** range(0,random.randrange(0,100))]  common=set()  **for** i **in** a:  **for** j **in** b:  **if**(i==j):  common.add(i)   print(a) print(b) print(common)   1. Write this in one line of Python (don’t worry if you can’t figure this out at this point - we’ll get to it soon)   **import** random a = [random.randrange(0,100) **for** i **in** range(0,random.randrange(0,100))]  b = [random.randrange(0,100) **for** i **in** range(0,random.randrange(0,100))]  common=set(i **for** i **in** a **for** j **in** b **if**(i==j))    print(a) print(b) print(common) |
| Ask the user for a string and print out whether this string is a palindrome or  not. (A **palindrome** is a string that reads the same forwards and backwards.)  x=input(**"enter a string:"**) y=x[::-1] **if**(x==y):  print(**"the given string is palindrome"**) **else**:  print(**"not a palindrome"**) |  |
| 4 | Let’s say I give you a list saved in a variable: a = [1, 4, 9, 16, 25, 36, 49, 64, 81, 100]. Write one line of Python that takes this list a and makes a new list that has only the even elements of this list in it.  a = [1, 4, 9, 16, 25, 36, 49, 64,81, 100] b = [x **for** x **in** a **if** x%2==0] print(b) |  | 2 |
| Make a two-player Rock-Paper-Scissors game. (Hint: Ask for player plays (using input), compare them, print out a message of congratulations to the winner, and ask if the players want to start a new game)  Remember the rules:  Rock beats scissors Scissors beats paper Paper beats rock  **def** func (k,l):  **if**(k==**"ROCK" and** l==**"SCISSOR"**)**or**(k==(**"SCISSOR"**)**and** l==(**"PAPER"**))**or**(k==(**"PAPER"**)**and** l==(**"ROCK"**)):  **return True  else**:  **return False** x=input(**"player 1 enter your choice"**) y=input(**"player 2 enter your choice"**)  a=x.upper() b=y.upper()  **if not**(a==**"ROCK" or** a==**"SCISSOR" or** a==**"PAPER" or** b==**"ROCK" or** b==**"SCISSOR" or** b==**"PAPER"**):  print(**"INVALID INPUT BY ANYONE"**)  **elif** a==b:  print(**"it's a tie"**)   **elif** func(a,b):  print(**"Player A wins"**)  **else**:  print(**"Player B wins"**) |  |

|  |  |  |  |
| --- | --- | --- | --- |
|  | Generate a random number between 1 and 9 (including 1 and 9). Ask the user to guess the number, then tell them whether they guessed too low, too high, or exactly right. (Hint: remember to use the user input lessons from the very first practical)  **import** random x=random.randrange(1,10) i=0; flag=**True while**(flag **and** (i<10)):  print(str(10-i)+**" chances remaining"**)  i=i+1  y=input(**"enter a number:"**)  **if** (int(y)==x):  flag=**False**;  print(**"you made correct guess CONGRATULATION"**)   **elif**(int(y)>x):  print(**"you have guessed too high, try lower number"**)   **else**:  print(**"you have guessed too low, try greater number"**) | 1. Keep the game going until the user types “exit”   **import** random x=random.randrange(1,10) i=0; flag=**True while**(flag):   i=i+1  y=input(**"enter a number:"**)  **if**(y==**"EXIT"**):  **break  if** (int(y)==x):  flag=**False**;  print(**"you made correct guess CONGRATULATION"**)   **elif**(int(y)>x):  print(**"you have guessed too high, try lower number"**)   **else**:  print(**"you have guessed too low, try greater number"**)   print(**"total number of guesses is"**+str(i))   1. Keep track of how many guesses the user has taken, and when the game ends, print this out.   **import** random x=random.randrange(1,10) i=0; flag=**True while**(flag):   i=i+1  y=input(**"enter a number:"**)  **if**(y==**"EXIT"**):  **break  if** (int(y)==x):  flag=**False**;  print(**"you made correct guess CONGRATULATION"**)   **elif**(int(y)>x):  print(**"you have guessed too high, try lower number"**)   **else**:  print(**"you have guessed too low, try greater number"**)   print(**"total number of guesses is"**+str(i)) |  |
| 5 | This week’s exercise is going to be revisiting an old exercise (see Practical 2), except require the solution in a different way.  Take two lists, say for example these two: a = [1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89]  b = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13]  and write a program that returns a list that contains only the elements that are common between the lists (without duplicates). Make sure your program works on two lists of different sizes. Write this in one line of Python using at least one list comprehension  a = [1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89] b = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13] print(set(i **for** i **in** a **for** j **in** b **if** i==j)) **"""LIST COMPREHENSIVE """** | The original formulation of this exercise said to write the solution using one line of Python, but a few readers pointed out that this was impossible to do without using sets that I had not yet discussed on the blog, so you can either choose to use the original directive and read about the set command in Python 3.3, or try to implement this on your own and use at least one list comprehension in the solution.  Extra:  Randomly generate two lists to test this  **import** random a = [random.randrange(0,100) **for** i **in** range(0,random.randrange(0,100))]  b = [random.randrange(0,100) **for** i **in** range(0,random.randrange(0,100))]  common=[i **for** i **in** a **for** j **in** b **if**(i==j)] sett=[] **for** i **in** common:  **if**(sett.\_\_contains\_\_(i)==**False**):  sett.append(i)   print(a) print(b) print(sett) | 2 |
| Ask the user for a number and determine whether the number is prime or not. (For those who have forgotten, a prime number is a number that has no divisors.). You can (and should!) use your answer to Practical 2 to help you.  Take this opportunity to practice using functions, described below.  x=input(**"enter a number"**)  flag=**True   for** i **in** range (2,int(int(x)/2)):  **if**(int(x)%i==0):  flag=**False  break  if** flag==**True**:  print(**"x is a prime number"**) **else**:  print(**"x is not a prime number"**) |  |
| Write a program that takes a list of numbers (for example, a = [5, 10, 15, 20, 25]) and makes a new list of only the first and last elements of the given list. For practice, write this code inside a function.  **def** firstandlast(x):  **return**([x[0],x[len(x)-1]]) a = [5, 10, 15, 20,25] print(firstandlast(a)) |  |
| 6 | Write a program that asks the user how many Fibonnaci numbers to generate and then generates them. Take this opportunity to think about how you can use functions. Make sure to ask the user to enter the number of numbers in the sequence to generate.(Hint: The Fibonnaci seqence is a sequence of numbers where the next number in the sequence is the sum of the previous two numbers in the sequence. The sequence looks like this: 1, 1, 2, 3, 5, 8, 13, …)  **def** fib(a,b,n):  c=a+b  print(c,end=**" "**)  **if**(n>0):  fib(b,c,n-1)   x=input(**"enter a number :"**) print(1,end=**" "**) **if**(int(x)>1):  print(1,end=**" "**) **if**(int(x)>2):  fib(1,1,int(x)-3) |  | 2 |
| Write a program (function!) that takes a list and returns a new list that contains all the elements of the first list minus all the duplicates.  a = [1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89] b = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13] customlist = set(i **for** i **in** a **for** j **in** b **if** i==j)  print(customlist) | Write two different functions to do this - one using a loop and constructing a list, and another using sets.  Go back and do Practical 2 using sets, and write the solution for that in a different |
| Write a program (using functions!) that asks the user for a long string containing multiple words. Print back to the user the same string, except with the words in backwards order. For example, say I type the string:  My name is Michele  Then I would see the string:  Michele is name My shown back to me.  **def** reverse(s):  y=s.split()  k=**""  for** i **in** reversed(y):  k=k+(i+**" "**)   **return** k  k=[] **for** i **in** range(0,1000000000):  k.append(float(i))  print(k) x=input(**"enter a long string"**) print(reverse(x)) |  |
|  | Write a password generator in Python. Be creative with how you generate passwords - strong passwords have a mix of lowercase letters, uppercase letters, numbers, and symbols. The passwords should be random, generating a new password every time the user asks for a new password. Include your run-time code in a main method.  **import** string **from** random **import** \* characters = string.ascii\_letters + string.punctuation + string.digits password = **""**.join(choice(characters) **for** x **in** range(randint(8, 16))) print(password) | Ask the user how strong they want their password to be. For weak passwords, pick a word or two from a list. |  |
| Use the BeautifulSoup and requests Python packages to print out a list of all  the article titles on the New York Times homepage.  *#web scrapping with python* **import** re **from** bs4 **import** BeautifulSoup **import** requests  result=requests.get(**"http://www.charusat.ac.in"**) print(result.status\_code) print(result)  c=result.content   soup=BeautifulSoup(c,features=**"lxml"**)  *#print (soup.prettify())* **for** link **in** soup.find\_all(**'a'**):  print(link.get(**'href'**)) |  |

|  |  |  |  |
| --- | --- | --- | --- |
| 7 | Create a program that will play the “cows and bulls” game with the user. The game works like this:  Randomly generate a 4-digit number. Ask the user to guess a 4-digit number. For every digit that the user guessed correctly in the correct place, they have a “cow”. For every digit the user guessed correctly in the wrong place is a “bull.” Every time the user makes a guess, tell them how many “cows” and “bulls” they have. Once the user guesses the correct number, the game is over. Keep track of the number of guesses the user makes throughout teh game and tell the user at the end.  Say the number generated by the computer is 1038. An example interaction could look like this:  Welcome to the Cows and Bulls Game! Enter a number:  >>> 1234  2 cows, 0 bulls  >>> 1256  1 cow, 1 bull  ...  Until the user guesses the number.  **import** random   **def** numberofmatch(randomnumber,x):  firstdigit = (randomnumber // 1000) % 10  seconddigit = (randomnumber // 100) % 10  thirddigit = (randomnumber // 10) % 10  fourthdigit = (randomnumber // 1) % 10  firstdigitguessed = (x // 1000) % 10  seconddigitguessed = (x // 100) % 10  thirddigitguessed = (x // 10) % 10  fourthdigitguessed = (x // 1) % 10  count=0  **if**(firstdigit==firstdigitguessed):  count+=1  **if**(seconddigit==seconddigitguessed):  count+=1  **if**(thirddigit==thirddigitguessed):  count+=1  **if**(fourthdigit==fourthdigitguessed):  count+=1  **return** count  print(**"WELCOME TO COWS AND BULL GAME"**) print(**'rules are :For every digit that the user guessed correctly in the correct place,they have a “cow”.\n For every digit the user guessed correctly in the wrongplace is a “bull.” Every time the user makes a guess, tell them how many“cows” and “bulls” they have.\n Once the user guesses the correct number,the game is over'**) randomnumber=random.randrange(1000,9999) print(randomnumber)  flag=**True while** flag:  x=int(input(**"enter a guess of four digit "**))  c=numberofmatch(randomnumber,x)  print(**"{} cows,{} bulls"**.format(c,4-c))  **if**(c==4):  flag=**False** print(**"congratulation you guessed correctly."**) |  | 2 |
| 8 | Using the requests and BeautifulSoup Python libraries, print to the screen the full text of the article on this website: http://www.vanityfair. com/society/2014/06/monica-lewinsky-humiliation-culture.  The article is long, so it is split up between 4 pages. Your task is to print out the text to the screen so that you can read the full article without having to click any buttons. This will just print the full text of the article to the screen. It will not make it easy to read, so next exercise we will learn how to write this text to a .txt file.  **import** requests **from** bs4 **import** BeautifulSoup **import** codecs **import** textwrap   **def** get\_web\_page(url):  r = requests.get(url)  **return** r.text  **def** format\_text\_write(content, fileName):  dedented\_text = textwrap.dedent(content).strip()  writeBuffer = textwrap.fill(dedented\_text, width = 100)  write\_text(fileName, writeBuffer)  **def** write\_text(text\_name, content):  f = codecs.open(text\_name,**'a'**)  f.write(content)   url = **'http://www.vanityfair.com/society/2014/06/monica-lewinsky-humiliation-culture'** fileName = **'VanityFair.txt'** html\_fileName = **'VanityFair\_html.txt'** FormattedFileName = **'VanityFair\_Formatted.txt'** html = get\_web\_page(url) soup = BeautifulSoup(html, **'html.parser'**, from\_encoding=**'utf-8'**) article = soup.find\_all(class\_=**"content-section"**)  **for** section **in** article:  paragraph = section.find\_all(**'p'**)  **for** sentence **in** paragraph:  **if** sentence **is not None**:  sentence\_text = sentence.get\_text()  **if** len(sentence\_text):  print(sentence\_text.encode(**'ascii'**, **'ignore'**))  write\_buffer = sentence\_text + **'\n'** write\_text(fileName, write\_buffer)  **with** open(fileName, **'r'**) **as** content\_file:  content = content\_file.read()  format\_text\_write(content, FormattedFileName) |  | 2 |
| Write a function that takes an ordered list of numbers (a list where the elements are in order from smallest to largest) and another number. The function decides whether or not the given number is inside the list and  returns (then prints) an appropriate boolean.  **def** contains(x,y):  **for** i **in** x:  **if**(i==y):  **return True  return False** x=input(**"enter few numbers for the list"**) y=list(x.split(**" "**)) z=[] print(y) **for** i **in** y:  z.append(int(i)) z.sort() print(z) inp=input(**"input any number"**) print(contains(x,inp)) | Use Binary Search |
| 9 | Take the code from the How To Decode A Website exercise , and instead of printing the results to a screen, write the results to a txt file. In your code, just make up a name for the file you are saving to.  **from** bs4 **import** BeautifulSoup **as** soup **import** requests **import** re  my\_url = **'https://www.nytimes.com'** *# opening connection, grabbing the page.* r = requests.get(my\_url) page\_html = r.text  *# html parsing* page\_soup = soup(page\_html, **'html.parser'**)   *# grabs each story header and stores into a list* stories = page\_html  print(**"Total number of stories: "** + str(len(stories)) + **". \n"**)  open\_file = open(**'file\_to\_save.txt'**, **'w'**)  *# Regex to eliminate tags and characters. Lstrip and rstrip to remove whitespaces/newlines.* **for** b **in** stories:  content = str(re.sub(**"<.\*?>"**, **""**, str(b))).lstrip().rstrip()  open\_file.write(**'\n'** + content)  open\_file.close() | Ask the user to specify the name of the output file that will be saved. | 2 |
| Given a .txt file that has a list of a bunch of names, count how many of each name there are in the file, and print out the results to the screen.  count=dict()  **with** open(**"jainil"**,**'r'**) **as** f:  x=f.read()  y=x.split()  **for** i **in** y:  count[i]=0  **for** i **in** y:  count[i]+=1  print(count) | Instead of using the .txt file from above (or instead of, if you want the challenge), take this .txt file, and count how many of each “category” of each image there are. This text file is actually a list of files corresponding to the SUN database scene recognition database, and lists the file directory hierarchy for the images. Once you take a look at the first line or two of the file, it will be clear which part represents the scene category. To do this, you’re going to have to remember a bit about string  parsing in Python 3. I talked a little bit about it in this post. |
| 10 | Develop programs to understand the control structures of python  **for** i **in** range(10):  print(i)  **if**(i==2):  **continue  if**(i==8):  **break  else**:  **pass** print(**"loop complete"**)  print() print()  k=0 **while True**:  k+=1  print(k)  **if**(k==3):  **continue  elif**(k==5):  **pass  elif**(k==9):  **break** print(**"loop complete"**) |  | 2 |
| Develop programs to learn different types of structures (list, dictionary, tuples)  thislist = [**"apple"**, **"banana"**, **"cherry"**] print(thislist) print(thislist[1]) thislist[1] = **"blackcurrant"** print(thislist) thislist = [**"apple"**, **"banana"**, **"cherry"**] **for** x **in** thislist:  print(x) **if "apple" in** thislist:  print(**"Yes, 'apple' is in the fruits list"**) print(len(thislist)) thislist.append(**"orange"**) print(thislist) thislist.insert(1, **"orange"**) print(thislist) thislist.remove(**"banana"**) print(thislist) thislist.pop() print(thislist) **del** thislist[0] print(thislist) thislist.clear() print(thislist) thislist = list((**"apple"**, **"banana"**, **"cherry"**)) *# note the double round-brackets* print(thislist)  thistuple = (**"apple"**, **"banana"**, **"cherry"**) print(thistuple) print(thistuple[1]) *# The values will remain the same:* print(thistuple) **for** x **in** thistuple:  print(x) **if "apple" in** thistuple:  print(**"Yes, 'apple' is in the fruits tuple"**) print(len(thistuple)) thistuple = tuple((**"apple"**, **"banana"**, **"cherry"**)) *# note the double round-brackets* print(thistuple) thisset = {**"apple"**, **"banana"**, **"cherry"**} print(thisset) **for** x **in** thisset:  print(x) thisset = {**"apple"**, **"banana"**, **"cherry"**} print(**"banana" in** thisset) thisset.add(**"orange"**) print(thisset) thisset.update([**"orange"**, **"mango"**, **"grapes"**]) print(thisset) print(len(thisset)) thisset.remove(**"banana"**) print(thisset) thisset.discard(**"banana"**) print(thisset) thisset = {**"apple"**, **"banana"**, **"cherry"**} x = thisset.pop() print(x) print(thisset) thisset.clear() print(thisset) thisset = set((**"apple"**, **"banana"**, **"cherry"**)) *# note the double round-brackets* print(thisset) thisdict = {  **"brand"**: **"Ford"**,  **"model"**: **"Mustang"**,  **"year"**: 1964 } print(thisdict) x = thisdict[**"model"**] thisdict[**"year"**] = 2018 **for** x **in** thisdict:  print(x) **for** x **in** thisdict:  print(thisdict[x]) **for** x **in** thisdict.values():  print(x) **for** x, y **in** thisdict.items():  print(x, y) **if "model" in** thisdict:  print(**"Yes, 'model' is one of the keys in the thisdict dictionary"**) print(len(thisdict)) thisdict[**"color"**] = **"red"** print(thisdict) thisdict.pop(**"model"**) print(thisdict) |
| Develop programs to understand working of exception handling and assertions  fh=open(**"testfile"**, **"w"**) **try**:  fh = open(**"testfile"**, **"w"**)  fh.write(**"This is my test file for exception handling!!"**) **except** IOError:  print (**"Error: can\'t find file or read data"**) **else**:  print (**"Written content in the file successfully"**)  **finally**:  fh.close() |
| Develop programs to learn concept of functions scoping, recursion and list mu  l=[] **def** a(n):  **if**(n<20):   l.append(n)  n+=1  a(n)  **return** l  **else**:  **return** print(a(1))  t |

|  |  |  |  |
| --- | --- | --- | --- |
| 11 | Develop programs to understand working of exception handling and assertions  fh=open(**"testfile"**, **"w"**) **try**:  fh = open(**"testfile"**, **"w"**)  fh.write(**"This is my test file for exception handling!!"**) **except** IOError:  print (**"Error: can\'t find file or read data"**) **else**:  print (**"Written content in the file successfully"**)  **finally**:  fh.close() | Recalling all concepts of above practicals | 2 |
| Develop programs for data structure algorithms using python – searching,  sorting and hash tables |
| Develop programs to learn regular expressions using python. |
| Develop chat room application using multithreading. |
| 12 | Learn to plot different types of graphs using PyPlot. | 2 |
| Implement classical ciphers using python. |
| 13 | Draw graphics using Turtle. |  | 2 |
| Develop programs to learn GUI programming using Tkinter. |
| 14 | Django Framework | 2 |
| 15 | Project Presentation |  | 2 |