**DAILY ASSESSMENT FORMAT**

|  |  |  |  |
| --- | --- | --- | --- |
| **Date:** | **23/may/2020** | **Name:** | **Akshay** |
| **Course:** | **python** | **USN:** | **4al17ec008** |
| **Topic:** | **Python Exercise** | **Semester & Section:** | **6th A** |
| **Github Repository:** | **Akshay-Online-Course** |  |  |

|  |
| --- |
| **FORENOON SESSION DETAILS** |
| **Image of session** |
| Report – Report can be typed or hand written for up to two pages.  Write python code to verify user\_name = "Micheal" and password ="e3$WT89x". The total number of attempts are 03. For every wrong user\_name  user\_name=input("Enter the username: ")  password =input("Enter the password :")  k=0  while k<3:  if user\_name =="Micheal" and password =="e3$WT89x":  print("You have successfully logged in")  else:  k+=1  print("Invalid username and password")  if k==3:  print("Account Locked") |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Date:** | **23/may/2020** | **Name:** | **Akshay** | |
| **Course:** | **Python** | **USN:** | **4al17ec008** | |
| **Topic:** | **Python Exercise** | **Semester&Section:** | **6th A** | |
| **Git hub repository** | **Akshay-Online-Course** |  |  | |
| **AFTERNOON SESSION DETAILS** | | | |
| **Image of session** | | | |
| Report – Report can be typed or hand written for up to two pages.  Conditional Expression  The order of the arguments is diﬀerent from many other languages (such as C, Ruby, Java, etc.), which may lead to bugs when people unfamiliar with Python's "surprising" behaviour use it (they may reverse the order). Some ﬁnd it "unwieldy", since it goes contrary to the normal ﬂow of thought (thinking of the condition ﬁrst and then the eﬀects.  if, elif, and else:  number = 5  if number > 2:  print("Number is bigger than 2.")  elif number < 2:  # Optional clause (you can have multiple elifs)  print("Number is smaller than 2.")  else: # Optional clause (you can only have one else)  print("Number is 2.")  Boolean Logic Expressions:  Boolean logic expressions, in addition to evaluating to True or False, return the value that was interpreted as True or False. It is Pythonic way to represent logic that might otherwise require an if-else test.  And operator  The and operator evaluates all expressions and returns the last expression if all expressions evaluate to True. Otherwise it returns the ﬁrst value that evaluates to False:  Or operator  The or operator evaluates the expressions left to right and returns the ﬁrst value that evaluates to True or the last value (if none are True)  Break and Continue in Loops:  break statement  #When a break statement executes inside a loop, control ﬂow "breaks" out of the loop immediately:  i = 0  while i < 7:  print(i)  if i == 4:  print("Breaking from loop")  break  i += 1  The loop conditional will not be evaluated after the break statement is executed. Note that break statements are only allowed inside loops, syntactically. A break statement inside a function cannot be used to terminate loops that called that function  continue statement  A continue statement will skip to the next iteration of the loop bypassing the rest of the current block but continuing the loop. As with break, continue can only appear inside loops  for i in (0, 1, 2, 3, 4, 5):  if i == 2 or i == 4:  continue  print(i)  0  1  3  5  Nested Loops  If you have a loop inside a function, using return from inside that loop is equivalent to having a break as the rest of the code of the loop is not executed (note that any code after the loop is not executed either):  def break\_loop():  for i in range(1, 5):  if (i == 2):  return(i)  print(i)  return(5)  #If you have nested loops, the return statement will break all loops:  def break\_all():  for j in range(1, 5):  for i in range(1,4):  if i\*j == 6:  return(i)  print(i\*j) | | | |