Python- Exercise03

**PYTHON DICTIONARY**

Dictionaries are similar to hash table type. They have pairs of key-value. Keys are unique within a dictionary. Keys can be data type – strings, numbers or tuples. Values can be of any data type. Keys are separated by ( : ). The items in the dictionary are separated by ( , ). Getting a value: using [ ] along with the key.

myDict = {empName: 'Simon', title: ‘Director’, yOfEmp: 8}

print ("myDict [empName]: ", myDict [empName])

print ("myDict [title]: ", myDict [title])

print ("myDict [Director]: ", myDict [Director]) ----------- ?

**Two important properties of keys:**

1. Duplication is not permitted meaning no more than one value is assigned to the same key. If it happens, the last assignment is considered.
2. Keys are immutable; they accept data types such as string, integer or tuples.

myDict = {[‘empName’]: 'Simon', ‘title’: ‘Director’, ‘yOfEmp’: 8} ------------------ ?

myDict = {‘empName’: 'Simon', ‘title’: ‘Director’, ‘yOfEmp’: 8, ‘title’:’ teacher’} ----------- ?

**Deleting Dictionary**

Three tasks can be performed: 1) removing individual elements 2) clearing a whole dictionary 3) deleting an entire dictionary.

myDict = {‘empName’: 'Simon', ‘title’: ‘Director’, yOfEmp: 8}

del myDict ['empName'] # remove entry with key 'Name' myDict.clear() # remove all entries in dict del myDict # delete entire dictionary

**Built-in Functions and Methods**

**Len (dict):** returns the length of dictionary (equal to the number of elements).

**Str (dict):** returns the string representation of a dictionary.

**Type (dict):** returns the type of python data structure.

**dict.copy():** returns a copy of a dictionary.

**dict.get (key, default value):** returns a value of a key or default if key is not in the dictionary.

**dict.items():** returns a list of (key, value) tuples.

**dict.keys():** returns a list of all keys of a dictionary.

**Dict.update(dict2)**: adds dictionary dict2 to dict1.

**Dict.values():** returns a list of all values of a dictionary.

**Extra examples**

**Sorting:**

dict = {'a2':'3', 'k1':'89', 'z4':'5', 'd3':'0'}

sortedDict = sorted(dict.items())

for k,v in sortedDict:

print (k, v)

**Searching for a key:**

list = {'Nabil':16,'Tanya':19}

search = int(input('enter the age you search for'))

for name, age in list.items():

if age == search:

print (name)

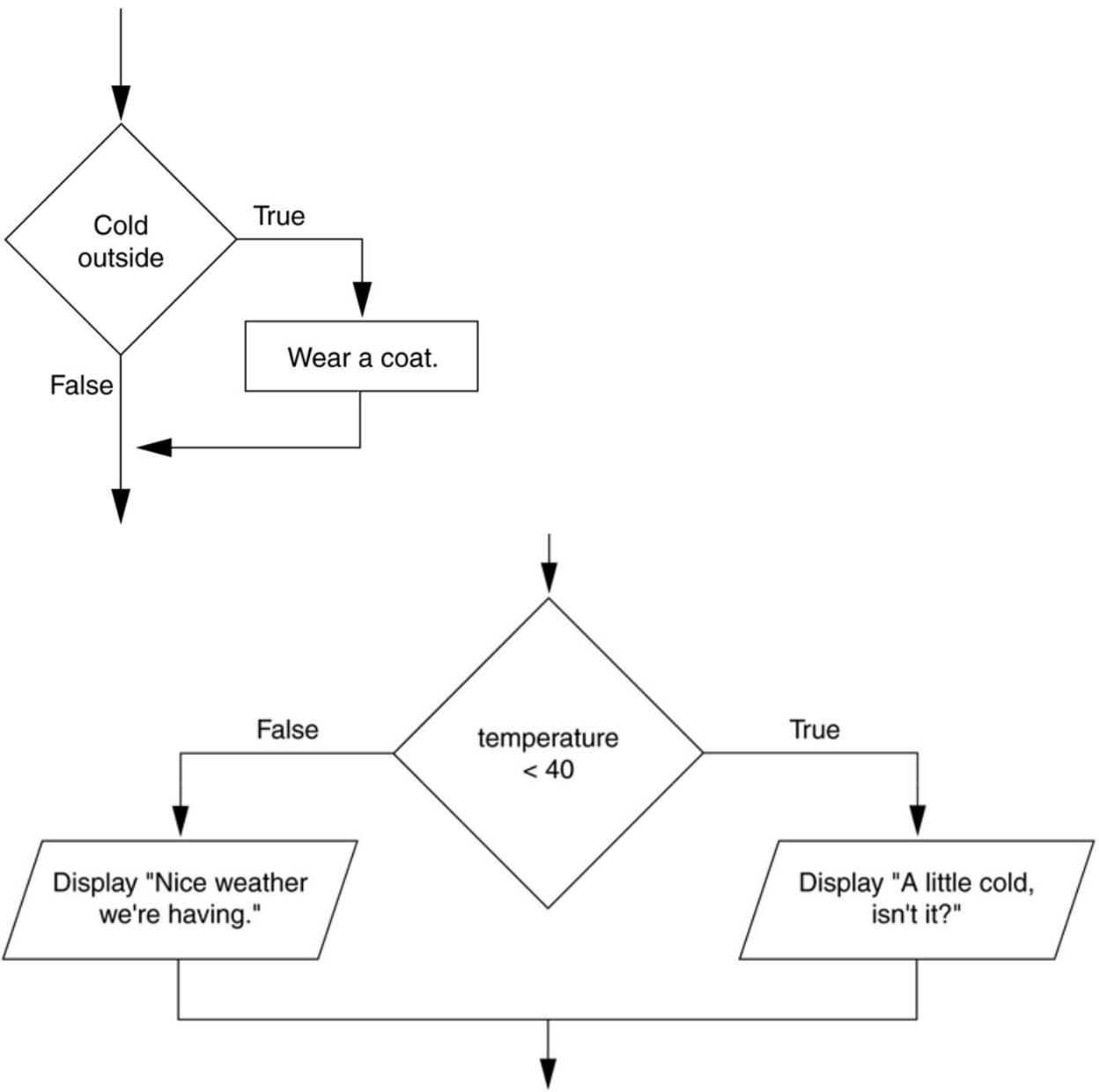
else:

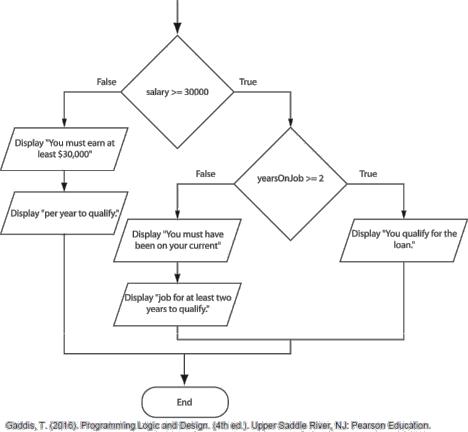
print ('key is not found')

**DECISION STRUCTURE (DS)**

DS allows a program to perform actions under certain conditions. Below (in the first figure), the program follows one path if a condition is true and leads to an action to perform. The program follows another path that skips that action if a condition is false. In the second figure, there are two possible paths of execution- one is taken if a condition is true and the other is taken if the condition is false. In the third figure, the decisions are nested in order to test more than one condition.

In Python, the True is equal to non zero or non null; False is equal to zero or null value.





**If …… Structure**

*if condition :*

indented Statement Block

weight = float(input("How many pounds does your suitcase weigh? "))

if weight > 50:

print("There is a $25 charge for luggage that heavy.")

print("Thank you for your business.")

**If …..Else …..Structure**

*if condition :*

indented Statement Block -True Condition

*else:*

indented Statement Block - False Condition

**If …..Else If…..Structure**

*if condition\_1:*

statement

statement

etc.

*elif* ***condition\_2***:

statement

statement

etc.

*Insert as many elif clauses as necessary…*

*else:*

statement

statement

etc.

**Nested *If...Elif...Else***

if expression1:

statement(s)

if expression2:

statement(s)

elif expression3:

statement(s)

else:

statement(s)

elif expression4:

statement(s)

else:

statement(s)

if totalHours <= 40:

totalWages = hourlyWage\*totalHours

else:

overtime = totalHours - 40

totalWages = hourlyWage\*40 + (1.5\*hourlyWage)\*overtime

print (“The total wage is” + totalWages)

dueAmount = Iint(input(“Please, emter the due amount: “)

if dueAmount < 1000:

interest= dueAmount \* 0.07

print ("the interest on”, dueAmount, “is:”, interest)

var2 = 0

elif dueAmount <10000:

interest= dueAmount \* 0.15

print ("the interest on”, dueAmount, “is:”, interest)

else:

interest= dueAmount \* 0.25

print ("the interest on”, dueAmount, “is:”, interest)

print (“See You Later”)

salary=int(input("enter a salary"))

yearOnJob=int(input(“enter years of experience”)

if salary>=3000:

if yearOnJob>=3:

print ("You qualify for a loan")

else:

print ("You qualify for a half of a loan")

else:

if yearOnJob>=10:

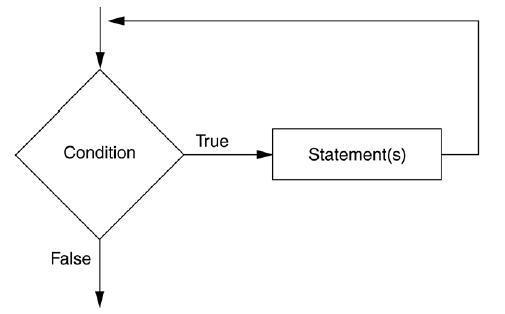
print ("You qualify for a half of a 25%% of a loan ")

else:

print ("You do not qualify for loans")

**Loops:**

A repetition structure causes a statement or set of statements to execute multiple times until this repetition is stopped using a special value.



Gaddis, T. (2016). Programming Logic and Design. (4th ed.). Upper Saddle River, NJ: Pearson Education.

**While loop**

Repeatedly executes a [block of] statement[s] as long as a condition is true (non-zer0, non-null).

**while Boolean expression:**

**Statement[s]**

Python uses indentation as its method of grouping statements.

**amount =0**

**while (amount <= 10000 and count < 10):**

**amount+= 100**

**count=+1**

**print (“ This was an example of while!”)**

**For … In**

It iterates items in a sequence until the entire sequence is exhausted. For each iteration, a statement or a block of statements is executed.

**for** *iterating variable* **in** sequence**:**

**statements(s)**

**# traversal of a string sequences**

**#JAVA FOR LOOP**

For (int i=0; i<10; i++)

{

System.out.printline(i);

System.out.printline(i\*i);

}

System.out.printline(“thank you”);

# PYTHON FOR LOOP

**# traversal of a string sequences**

**for letter in 'English':**

**print ('The current Letter is :', letter)**

**print(‘/n’)**

colors = [‘yellow’, 'red', 'blue',’black’]

for color in colors: # traversal of List sequence

print ('The current color is :', color)

print ("This is an example of for in structure!")

**Try** iterating and printing elements of a list li = ['a', 'b', 'c', 'd', 'e'] using their indices.

**For .. Else Structure**

The **else** statement can be used with a **for** loop; however, the **else** block is executed only if the **for** loop terminates normally (not by a break statement).

for letter in 'English': # traversal of a string sequence if letter =='i':

print ('there is letter',letter)

break

else: print ('we can\'t find a match letter')

**While .. Else Structure**

The **else** statement is also used with a **while** loop. The **else** statement is executed when the condition is false.

limit = 1000; interest = 0.1

balance = float (input(‘ enter a balance: ‘))

while balance < limit:

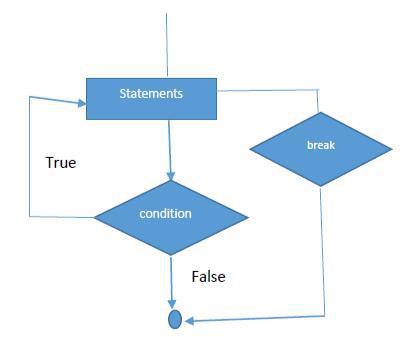
balance =balance + balance \* interest

print ('the balance is now: ', balance)

else: print (‘ balance is greater than limit’)

**Break Statement**

It is used to terminate a loop (iteration process) and resume execution of the next statement (outside the loop block). In case of a nested loop, break will cause the innermost loop to exit and resume the next statement outside the block.



for letter in 'French':

if letter == 'c':

break

print ('The current Letter :', letter)

In below, example, the 5th and 6th lines can terminate the while loop if the condition is true and then the 7th line is executed (print ("Good bye!"))

dig = 10

while dig > 0:

print ('The current value is:', dig)

dig = dig -1

if dig == 5:

break

print ("Thanks!")

**Continue Statement**

It skips the current execution of the iteration and continues to check the condition.

dig = 15

while dig > 0:

dig -= 1

if dig == 7:

continue

print ('The current value is:', dig)

print ("Thanks!")

**Iterator Object**

In Python, iterator object implements two methods **iter()** and **next()**. It traverses through elements using **for** or **while**.

Iterator can work on tuples, lists and strings.

**It = iter(list)** # this constructs an iterator object, where list is a pre-defined list.

**Print (next(it))** # this prints next available element of the iterator.

**More Programs to practice**

* **Multiple alternative decision making (IF ….ELSEIF ……)**

var = 100

if var == 200:

print ("1 - Got a true expression value")

print (var)

elif var == 150:

print ("2 - Got a true expression value")

print (var)

elif var == 100:

print ("3 - Got a true expression value")

print (var)

else:

print ("4 - Got a false expression value")

print (var)

print ("Good bye!")

* **Nested IF statements**

salary=int(input("enter a salary"))

yearOnJob=int(input("enter years of experience"))

if salary>=3000:

if yearOnJob>=3:

print ("You qualify for a loan")

else:

print ("You qualify for a half of a loan")

else:

if yearOnJob>=10:

print ("You qualify for a half of a 25% of a loan ")

else:

print ("You do not qualify for loans")

* **Looping inside a control statement**

amount =0

count = int(input("enter a number less than 10 "))

if count < 10:

while (count <10):

count+= 1

print ("count", count)

else:

print (" You entered an invalid value")

print ("This was another example of while!")

* **WHILE…ElSE inside FOR…ELSE**

for x in list(range (1,4)):

num = int(input ('enter a number'))

while num <= 10:

print ('the number ', num,' is good')

num+=1

else:

print ('the number ', num,' is not good')

else:

print ('sorry! No more chance')

* **NEXT function to point to next position in a string**

str = "formidable"

it = iter(str)

for x in it:

print (next(it))

* **Printing a list by traversing indices using *range* function.**

li = ['r', 'y', 'e', 'r', 's', 'o', 'n']

for i in range(len(li)):

print (li[i])

* **Complex Boolean expression (**if var == 'c' or var =='a')

var = input('enter something from the keyboard')

if var == 'c' or var =='a':

print ("1 - Got a true expression value")

print (var)

else:

print ('Thanks!')

* **Using continue (the looping process after skipping the current execution)**

dig = 15

while dig > 0:

dig -= 1

if dig == 7:

continue

print ('The current value is:', dig)

print ("Thanks!")

* **Using break (to cut the looping process and exit)**

dig = 10

while dig > 0:

print ('The current value is:', dig)

dig = dig -1

if dig == 5:

break

print ("Thanks!")