Introduction to Python

In [2]:

```
x = 5
y = 6
print(x**y)
```

```
15625
```

In [3]:

```
totalStudents = 35
totalMale = 20
totalFemale = totalStudents - totalMale
```

In [4]:

```
totalFemale
```

Out[4]:

```
15
```

Creating and filling an empty dictionary

In [5]:

```
custNames = ['Collins', 'Goodnews', 'Mary', 'Mark', 'Rachael']
acctBals = [5374.98, 75643.99, 65437.3, 436673.67,75643.99 ]
```

In [6]:

```
acctsBalDict = dict() # Or we can use "acctsBalDict = {}"

acctsBalDict['Collins'] = 5374.98
acctsBalDict['Goodnews'] = 75643.99
acctsBalDict['Mary'] = 65437.3
acctsBalDict['Mark'] = 436673.67
acctsBalDict['Rachael'] = 75643.99

```

In [7]:

```
acctsBalDict
```

Out[7]:

```
{'Collins': 5374.98,
 'Goodnews': 75643.99,
 'Mary': 65437.3,
 'Mark': 436673.67,
 'Rachael': 75643.99}
```

Scenario: we have a class of 5 students. we want to assign a bonus score to any student that achieves above 80% in the course exam

In [8]:

```
studNameList = ['Collins','Goodnews','Mary', 'Mark','Rachael']
examScoreList = [67.5, 95.0, 77.8, 90.3, 35.8]
```

Adding 10 points bonus to any student with score above 80 and ensuring student score does not exceed 100. Outputting the score results in a dictionary using the if and else statement

In [20]:

```
# Dictionary Implementation
'''
Dictionaries store information in key and Value pairs.
for example; student name as key and exam score as acssociated valkue
'''

studRecDict = {}
if len(studNameList) == len(examScoreList): # sanity check
    for stud, score in zip(studNameList,examScoreList):
        currentStudent = stud
        currentStdScore = score
 #print(currentStudent, currentStdScore)
 #Task: check student score to see if its above 80
        if currentStdScore > 80:
            currentStdScore = currentStdScore + 10
            if currentStdScore <= 100:
                studRecDict[currentStudent] = currentStdScore
            else:
                currentStdScore = 100
                studRecDict[currentStudent] = currentStdScore
        else:
            studRecDict[currentStudent] = currentStdScore
```

In [21]:

```
studRecDict
```

Out[21]:

```
{'Collins': 67.5, 'Goodnews': 100, 'Mary': 77.8, 'Mark': 100, 'Rachae
l': 35.8}
```

Outputting the score results in a List using an if and else statement

In [22]:

```
# List implementation
'''
Dictionaries store information in key and Value pairs.
for example; student name as key and exam score as acssociated valkue
'''

studScoreList = []
studList2 = []
if len(studNameList) == len(examScoreList): # sanity check
    for stud, score in zip(studNameList,examScoreList):
        currentStudent = stud
        currentStdScore = score
#print(currentStudent, currentStdScore)
#Task: check student score to see if its above 80
        if currentStdScore > 80:
            currentStdScore = currentStdScore + 10
            if currentStdScore <= 100:
                studList2.append(currentStudent)
                studScoreList.append(currentStdScore)

            else:
                currentStdScore = 100
                studList2.append(currentStudent)
                studScoreList.append(currentStdScore)
        else:
            studList2.append(currentStudent)
            studScoreList.append(currentStdScore)
```

In [23]:

```
print(studScoreList)
print(studList2)
```

```
[67.5, 100, 77.8, 100, 35.8]
['Collins', 'Goodnews', 'Mary', 'Mark', 'Rachael']
```

For Loop

In [24]:

```
studName = 'Patrick'
for char in studName:
 idx = studName.index(char)
 print('the chartarer :', char,'is in the position :', idx )
print('this is the end of the loop')
```

```
the chartarer : P is in the position : 0
the chartarer : a is in the position : 1
the chartarer : t is in the position : 2
the chartarer : r is in the position : 3
the chartarer : i is in the position : 4
the chartarer : c is in the position : 5
the chartarer : k is in the position : 6
this is the end of the loop
```

Do-While loop

In [25]:

```
listLen = len(custNames)
listLen
acctsBalDict2 = dict()
counter = 0
while counter < listLen:
 ctr = counter
 currentCust = custNames[ctr]
 bal = acctBals[ctr]
 acctsBalDict2[currentCust]= bal
 counter +=1
```

In [26]:

```
acctsBalDict2
```

Out[26]:

```
{'Collins': 5374.98,
 'Goodnews': 75643.99,
 'Mary': 65437.3,
 'Mark': 436673.67,
 'Rachael': 75643.99}
```

In [27]:

```
type(acctsBalDict2)
```

Out[27]:

```
dict
```

Functions

In [32]:

```python
# Creating a function to generate table of defaulting customersabs
import pandas as pd
def daysCompute(flagedCust):
# structure of parametrer is: customername, loan amount, days overdue
    custdf = pd.DataFrame(columns = ['custName','amount', 'Overdue'])

    cusnameL = []
    amt = []
    days = []
    for items in flagedCust:
        customer = items[0]
        loanAmt = items[1]
        dayODue = items[2]

        cusnameL.append(customer)
        amt.append(loanAmt)
        days.append(dayODue)

    custdf['custName'] = cusnameL
    custdf['amount'] = amt
    custdf['Overdue'] = days

    return custdf
```

In [34]:

```python
# Scenario1
custNames = ['Collins', 'Goodnews', 'Mary', 'Mark', 'Rachael']
loanBals = [5374.98, 75643.99, 65437.3, 436673.67,75643.99 ]
daysOverdue = [24, 65, 90, 170, 99]

#Requirement
# output a report of customers who are over 90 days due

overDueDaysMin = 90
flagedCust = []
for dayOver in daysOverdue:
    if dayOver > overDueDaysMin:
        flagIndx = daysOverdue.index(dayOver)
        cust = custNames[flagIndx]
        loan = loanBals[flagIndx]

#print(cust, loan, dayOver)
        flagedCust.append((cust, loan, dayOver))

# Now we call a funtion and pass the list of defaulting customer to it
        defaulttable = daysCompute(flagedCust)

flagedCust
```

Out[34]:

```
[('Mark', 436673.67, 170), ('Rachael', 75643.99, 99)]
```

In [35]:

```
defaulttable
```

Out[35]:

	custName	amount	Overdue
0	Mark	436673.67	170
1	Rachael	75643.99	99

Assigning grades to student score using functions

In [42]:

```
#Step 1: Create functionabs
# Statement Scope
def grader(score):
    grade = ''
    if score > 80:
        grade = 'A'
    elif score > 60 and score < 80:
        grade = 'B'
    elif score > 50 and score < 60:
        grade = 'C'
    else:
        grade = 'F'

    return grade

studNameList = ['Collins','Goodnews','Mary', 'Mark','Rachael']
examScoreList = [67.5, 95.0, 77.8, 90.3, 35.8]

for score in examScoreList:
    studentScore = score

# get the grade of the student by passing the score to the grade funtion
    grade = grader(score)

    print(score, grade)
```

```
67.5 B
95.0 A
77.8 B
90.3 A
35.8 F
```

In [46]:

```
# Advanced version
#Step 1: Create functionabs
# Statement Scope
def grader(score):
    grade = ''
    if score >= 80:
        grade = 'A'
    elif score >= 60 and score < 80:
        grade = 'B'
    elif score >= 50 and score < 60:
        grade = 'C'
    else:
        grade = 'F'

    return grade

studNameList = ['Collins','Goodnews','Mary', 'Mark','Rachael']
examScoreList = [67.5, 95.0, 77.8, 80, 35.8]

for student,score in zip(studNameList, examScoreList):
    studentScore = score
    name = student

# get the grade of the student by passing the score to the grade funtion
    grade = grader(studentScore)
    studentReport = [name, studentScore, grade]


    print(studentReport)
```

```
['Collins', 67.5, 'B']
['Goodnews', 95.0, 'A']
['Mary', 77.8, 'B']
['Mark', 80, 'A']
['Rachael', 35.8, 'F']
```

In [54]:

```
testList = [4,7,8, 10, 12]
print(testList[0])
print(testList[1])
print(testList[2])
print(testList[3])
print(testList[4])
```

```
4
7
8
10
12
```

In [55]:

```
for stan in testList:
    print(stan)
    y = stan*2
    print(y)
    print('-----')
```

```
4
8
-----
7
14
-----
8
16
-----
10
20
-----
12
24
-----
```

In [ ]:

```

```