COP 3035

Intro Programming in Python

Summer 2024

Lecture 12 – part 1

Exam 2 - 06/21/24
Lab 6 - 06/24/24
Homework 4 – 06/28/24

Lecture 12 – part 2

Review

Review

Integration exercise – Grade conversion

List comprehensions

Dictionary comprehensions

class.csv

	Student	Quiz1	Quiz2	Quiz3	Quiz4
1	S1	100	67	80	72
2	S2	89	70	78	90
3	S3	67	87	97	100
4	S4	78	90	65	98

Grade Conversion Table

Score	Letter	Score	Letter	Score	Letter	Score	Letter	Score	Letter
93-100	A	85-89	B+	75-79	B-	68-71	C	50-59	D
90-92	A-	80-84	B	72-74	C+	60-67	C-	0-49	F

Results

```
------------------------------------------------------------
Student        Quiz1      Quiz2      Quiz3      Quiz4

------------------------------------------------------------
S1               A          B          C+         C+
S2               B+         B-         A-         A-
S3               C-         A          A          A
S4               B-         C-         A          A
```

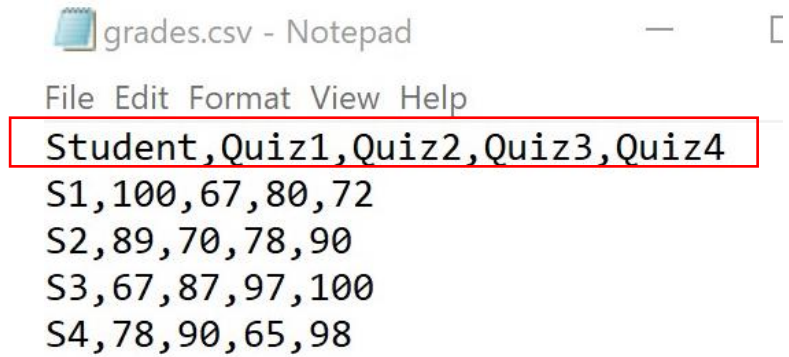
Plan:

1. Create a table of grade equivalences as a **dictionary**, **using tuples as keys for intervals**.
2. Open the file and store each line (representing a student) in a list of strings.
3. Iterate over the grade list; use enumerate() to obtain the index for each line.
4. For each grade, **check if it falls within a specified interval tuple**.
5. Append the corresponding grade equivalences to the result list.
6. Print the results.

```python
# Step 1 - Conversion table in a dictionary
d = {(93,100):"A",
     (90,92):"A-",
     (85,89):"B+",
     (80,84):"B",
     (75,79):"B-",
     (72,74):"C+",
     (68,71):"C",
     (60,67):"C-",
     (50,59):"D",
     (0,49):"F"}
```

```python
# Step 2 - Read student info from the file
with open('grades.csv','r') as filename:
    lines = []
    for l in filename:
        lines.append(l.strip())
```

```python
# Print the header
h = lines[0].split(',')
print('-'*60)
print(f'{h[0]:^10} {h[1]:^10} {h[2]:^10} {h[3]:^10} {h[4]:^10}')
print('-'*60)
```



lines[0]

```
------------------------------------------------------------
 Student    Quiz1      Quiz2      Quiz3      Quiz4
------------------------------------------------------------
```

```
S1,100,67,80,72
S2,89,70,78,90
S3,67,87,97,100
S4,78,90,65,98
```

lines[1]

```
for s in lines[1:]:  # Step 3 - Iteration for each student
    grades = s.split(',')
    s_letter = []
    for index, grade in enumerate(grades):  # Iterate over each quiz for each student
        if index !=0:
            grade_num = int(grade)
            for interval,letter in d.items():  # Iterate over the grade table
                if interval[0] <= grade_num <= interval[1]: # Step 4 - Check the interval
                    s_letter.append(letter) # Step 5
        else:
            s_letter.append(grades[0])  # Step 5
    # Step 6 - Print the results
    print(f'{s_letter[0]:^10} {s_letter[1]:^10} {s_letter[2]:^10} {s_letter[3]:^10} {s_letter[4]:^10}')

print('-'*60)
```

[S1,100,67,80,72]

[S1,100,67,80,72]

[S1,A,B,C+,C+] s_letter

[S1,A,B,C+,C+] s_letter

```
    S1          A          C-         B          C+
    S2          B+         C          B-         A-
    S3          C-         B+         A          A
    S4          B-         A-         C-         A
------------------------------------------------------------
```

List Comprehensions

- **List comprehensions** are a concise way to create lists in Python.
- They offer a shorter syntax for creating lists when compared to using loops.

Syntax form:

```
[expression for item in iterable]
```

```
[expression for item in iterable if condition]
```

for loop

```
myString = "Hello"

myList = []
for c in myString:
    myList.append(c)
print(myList)

['H', 'e', 'l', 'l', 'o']
```

Basic list comprehension

```
myList = [c for c in myString]
print(myList)

['H', 'e', 'l', 'l', 'o']
```

for loop

```
squares = []

for x in range(0,10):
    squares.append(x*x)

print(squares)

[0, 1, 4, 9, 16, 25, 36, 49, 64, 81]
```

Basic list comprehension

```
squares = [x*x for x in range(0,10)]

print(squares)

[0, 1, 4, 9, 16, 25, 36, 49, 64, 81]
```

Lecture 12 – part 3

Dictionary comprehensions

Dictionary Comprehensions

- **Like list comprehensions**, they offer a shorter syntax for creating dictionaries when compared to using loops.

Syntax form:

```
{key: value  for variable in iterable}

{key: value  for variable in iterable if condition}
```

Examples:

```
squares = {}
for x in range(6):
    squares[x] = x**2
Print(squares)

{0: 0, 1: 1, 2: 4, 3: 9, 4: 16, 5: 25}
```

```
squares = {x: x*x for x in range(6)}
print(squares)


{0: 0, 1: 1, 2: 4, 3: 9, 4: 16, 5: 25}
```

Examples:

```
values = ['apple', 'banana', 'cherry’]
for i, value in enumerate(values):
    dictionary[value] = i
print(dictionary)


{0: 'apple', 1: 'banana', 2: 'cherry'}
```

```
values = ['apple', 'banana', 'cherry']
dictionary = {i: value for i, value in enumerate(values)}
print(dictionary)



{0: 'apple', 1: 'banana', 2: 'cherry'}
```

Lecture 12 – part 4

Ternary operator

Ternary Operator

- The ternary operator in Python is a concise way to execute simple **if-else** statements in a **single line**.
- It is also known as the conditional expression.
- The basic syntax of the ternary operator is:

```
(a if condition else b)
```

```
x = 10
result = "Greater than 5" if x > 5 else "Less than or equal to 5"
print(result)

Greater than 5
```

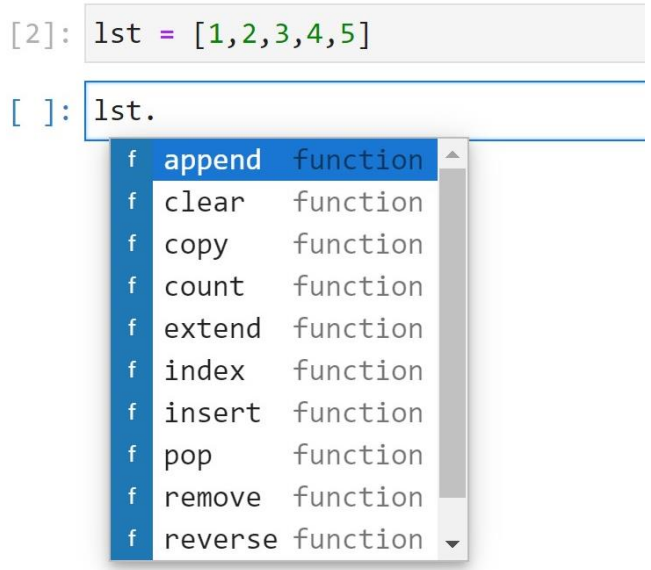
Lecture 12 – part 5

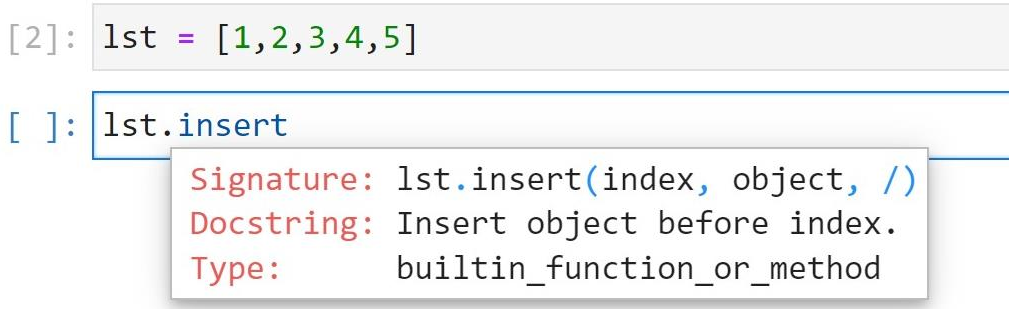
Functions

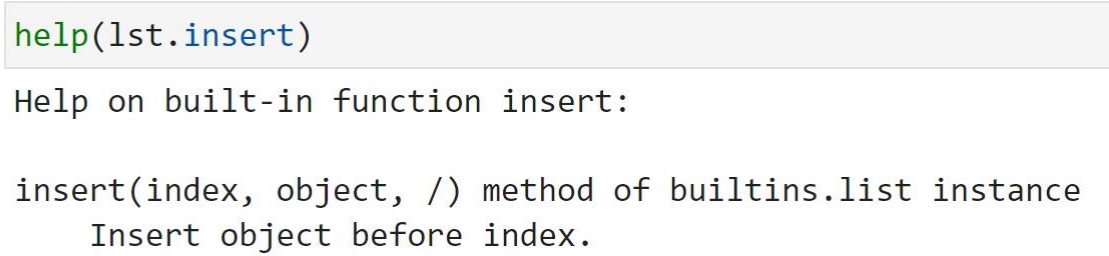
Built-in Methods

https://docs.python.org/

- Use Shift+Tab in the Jupyter Notebook to get more help about the method.
- You can also use the **help()** function:







What is a function ?

- A function is a valuable tool that groups a set of statements together, allowing them to be executed multiple times.
- This prevents us from having to write the same code repeatedly.

Function Syntax

```
def name_of_function(arg1,arg2):
    '''
    This is where the function's Document String (docstring) goes.
    When you call help() on your function it will be printed out.
    '''
    # Do stuff here
    # Return desired result
```