Operations with tables

Lists

Getting a range from a list

In
```
digits_names = [
"zero", "one", "two", "three", "four", "five", "six", "seven", "eight", "nine"
]

# list both margins of the range (right will not be included)
print(digits_names[4:7])

# drop the left margin in order to go from the beginning of the list
print(digits_names[:5])

# drop the right margin in order to go to the end of the list
print(digits_names[7:])
```

Out
```
['four', 'five', 'six']
['zero', 'one', 'two', 'three', 'four']
['seven', 'eight', 'nine']
```

Adding an element to the end of a list

In
```
emoji = ["Grinning", "Beaming", "ROFL", "Tears of Joy"]

print(emoji)

emoji.append("Winking")

print(emoji)
```

Out
```
['Grinning', 'Beaming', 'ROFL', 'Tears of Joy' ]
['Grinning', 'Beaming', 'ROFL', 'Tears of Joy', 'Winking' ]
```

Sorting tables (a list of lists) by column

In
```
data.sort(key=lambda row: row[1], reverse=True)
```

Loops

Getting a range of numbers or repeat the code

In
```
for element in range(5):
    print(element)
```

Out
```
0
1
2
3
4
```

In
```
for i in range(3):
    print("*****")
```

Out
```
*****
*****
*****
```

Modifying a list with a loop

In
```
...


for i in range(len(data)):
  part = data[i][1]/emojixpress_total
  data[i].append(part)


...
```

Aligning and displaying text to specifications

In
```
print("Emoji ", end='')
print("analysis")
```

Out
```
Emoji analysis
```

Glossary

Named argument
A function argument with a listed name

Lambda
Unnamed function with special syntax

Normalization
Converting values to a universal scale

Method
A function associated with a data type that works with a variable of that type. Methods are called using **dot notation**

List slice
Part of the list elements in a particular range of indices