python™

SOEN 287 WEB PROGRAMMING

PYTHON TOPICS

- About Python
- Python IDE
- First Program
- Strings
- Variables
- Lists
- For Loops
- Conditionals
- While Loops
- **Dictionaries**
- Comprehensions
- Files
- Exceptions
- Functions

DICTIONARY

- A dictionary is mutable and container type can store any number of python objects.
- Also known as *associative arrays* or hash (#) table
- Dictionaries consists of pairs (items) of keys and their corresponding values.
- The values of a dictionary can be of any type, but the keys must be a mutable data type such as strings, numbers or tuples.
- **Major difference** with lists: dictionaries are not **sorted**, meaning that (key, value) pairs are not kept in any specific order

DICTIONARIES EXAMPLE

In [3]:
```
data = {"age":88, "lastname": "Doe", "firstname": "John"}
print(data)
print(data["lastname"])
data["age"] = 66
print(data)
```

Output ?

```
{'age': 88, 'lastname': 'Doe', 'firstname': 'John'}
Doe
{'age': 66, 'lastname': 'Doe', 'firstname': 'John'}
```

DICTIONARIES EXAMPLE

- We can either remove or delete individual dictionary elements or clear the entire contents of a dictionary

```
In [12]: data = {"age":88, "lastname": "Doe", "firstname": "John"}
         del data["lastname"]
         print(data)
         data.clear()
         print(data)
         del data
         print(data)
```

Output ?

```
{'age': 88, 'firstname': 'John'}
{}

---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
<ipython-input-12-53dc657a9b5c> in <module>
      5 print(data)
      6 del data
----> 7 print(data)

NameError: name 'data' is not defined
```

LOOPING ON DICTIONARIES

```
data = {"age":66, "lastname": "Doe", "firstname": "John"}
```

In [38]:
```
for k in data:     # loop on keys
    print(k)
```

Output ?

```
age
lastname
firstname
```

In [39]:
```
for k in data:     # loop on keys, get associated values
    print(k, data[k])
```

Output ?

```
age 66
lastname Doe
firstname John
```

LOOPING ON DICTIONARIES

```
data = {"age":66, "lastname": "Doe", "firstname": "John"}
```

In [40]:
```
for k, v in data.items():    # loop on dictionary items
    print(k, v)
```

Output ?

```
age 66
lastname Doe
firstname John
```

In [41]:
```
for k in sorted(data):    # loop on sorted keys
    print(k, data[k])
```

Output ?

```
age 66
firstname John
lastname Doe
```

LOOPING ON DICTIONARIES

In [42]:
```
for k, v in sorted(data.items()):    # loop on sorted items
    print(k, v)
```

Output ?

```
age 66
firstname John
lastname Doe
```

In [43]:
```
# loop on reverse sorted items
for k, v in reversed(sorted(data.items())):
    print(k, v)
```

Output ?

```
lastname Doe
firstname John
age 66
```

PYTHON TOPICS

- About Python
- Python IDE
- First Program
- Strings
- Variables
- Lists
- For Loops
- Conditionals
- While Loops
- Dictionaries
- **Comprehensions**
- Files
- Exceptions
- Functions

LIST COMPREHENSIONS

- Build a list from another list, by transforming the elements
- Example: build a list of cube values

In [45]:
```
cubes = [x**3 for x in range(10)]
print(cubes)
```

Output ?

```
[0, 1, 8, 27, 64, 125, 216, 343, 512, 729]
```

LIST COMPREHENSIONS

- Example: build a list of cube values

```
In [45]: cubes = [x**3 for x in range(10)]
         print(cubes)
```

- the square brackets [] are used to create a list
- the elements are not listed explicitly: instead, they are computed from another list or sequence
  - in this case, we compute the cube of a number x**3
  - and we apply the computation to a sequence of numbers created with the **range** function
  - but it could be applied to any list already created

LIST COMPREHENSIONS

In [46]:
```
numbers = [4, -67, 12, 32, 14, -2, 7]
cubes = [x**3 for x in numbers]
print(cubes)
```

Output ? `[64, -300763, 1728, 32768, 2744, -8, 343]`

- We can also filter out some numbers
- For example, we might want to filter out negative values

In [47]:
```
numbers = [4, -67, 12, 32, 14, -2, 7]
cubes = [x**3 for x in numbers if x >= 0]
print(cubes)
```

Output ? `[64, 1728, 32768, 2744, 343]`

LIST COMPREHENSIONS

```
scores = [90, 52, 69, 100, 89, 78, 95, 75]
```

- Example: add 5 to all the scores

In [48]:
```
print(scores)
scores_plus5 = [x+5 for x in scores]
print(scores_plus5)
```

Output ?

```
[90, 52, 69, 100, 89, 78, 95, 75]
[95, 57, 74, 105, 94, 83, 100, 80]
```

LIST COMPREHENSIONS

- Example: avoiding getting scores above 100
- We could filter the list, but do not lose any scores

In [49]:
```
print(scores_plus5)
scores_plus5_max100 = [x for x in scores_plus5 if x <= 100]
print(scores_plus5_max100)
```

Output ?

```
[95, 57, 74, 105, 94, 83, 100, 80]
[95, 57, 74, 94, 83, 100, 80]
```

LIST COMPREHENSIONS

- We cannot put an else at the end of the list comprehension
- We have to move the if … else … at the beginning

In [50]:
```
scores_plus5_max100 = [x if x <= 100 else 100 for x in scores_plus5]
print(scores_plus5_max100)
```

Output ? `[95, 57, 74, 100, 94, 83, 100, 80]`

- The problem is that, in this way, we need 2 list comprehensions…

LIST COMPREHENSIONS

- Better to use a function in a single list comprehension to get the correct score immediately

In [51]:
```
print(scores)
scores_plus5 = [min(x+5, 100) for x in scores]
print(scores_plus5)
```

Output ?

```
[90, 52, 69, 100, 89, 78, 95, 75]
[95, 57, 74, 100, 94, 83, 100, 80]
```

DICTIONARY COMPREHENSIONS

- Same principle as list comprehensions, but build a dictionary instead
- Example: build a dictionary of cube values

In [52]:
```
numbers = [4, -67, 12, 32, 14, -2, 7]
cubes = {x: x**3 for x in numbers}
print(cubes)
```

Output ?

```
{4: 64, -67: -300763, 12: 1728, 32: 32768, 14: 2744, -2: -8, 7: 343}
```

DICTIONARY COMPREHENSIONS

In [17]:
```
for k in sorted(cubes):
    print(k, "**3=", cubes[k])
```

Output ?

```
-67 **3 = -300763
-2 **3 = -8
4 **3 = 64
7 **3 = 343
12 **3 = 1728
14 **3 = 2744
32 **3 = 32768
```

DICTIONARY COMPREHENSIONS

- In this particular case, it might be better to create a list of points
- In Python, the points created with parentheses () are called *tuples*
- Tuples are exactly like lists, except that they are immutable (they cannot change)

In [54]:
```
numbers = [4, -67, 12, 32, 14, -2, 7]
cubes = [(x, x**3) for x in sorted(numbers)]
print(cubes)
```

Output ?

```
[(-67, -300763), (-2, -8), (4, 64), (7, 343), (12, 1728), (14, 2744), (32, 327
68)]
```