04. Python Control Structure Loops

3ikakke

Outline

- Objectives
- Quick review of things we should know
- What are loops?
- While loops
- Iterables
- range() and len()
- For loops
- For loops showing indexes
- for loops with dictionaries
- Review of objectives
- Gist of the day
- Q&A

Objectives

- Understand loops
- Understand loop control with the Boolean True
- Understand infinite loops
- Understand iterables
- Understanding increments
- Mixing Logic and Loops
- Introducing new functions today
  - range()
  - len()

  - enumerate()

- Introducing a new method for dictionaries
  - dict.items()

Quick review of things we should know...

- Python datatypes
- Python operators
- Control Structure Logic
- How and why are these relevant?
- Some functions we now know
  - print
  - type

What are loops?

- Loops are - loops! Synonyms may include cycles.
- Repeated actions that continue till a condition stops them

While loops

- Very similar to English language.

```
i = 1
while i < 10:
    print(i)
    i = i + 1
```

While loops

- When making while loops rememnber to increment your counter!
- Else you may end up with a loop that never ends!!!
- Infinite loops are a bad thing.

Iterables

- Anything that you can iterate over is an iterable
- Examples
  - Lists
  - Tuples
  - Sets

  - Dictionaries
  - Others...
    * range(1, 10)

- Usually you can find how many items are in an iterable using the `len` function even if you cant index them
  - Sets
  - range()

range() and len()

- range() takes 1, 2 or 3 **arguments**
- Function arguments are what are passed into a function
- If only one argument it is assumed to be from 0 to that argument
  - range(10) implies range(0, 10, 1) and will return => **0, 1, 2, 3, 4, 5, 6, 7, 8, 9**
- If two arguments its is assumed to be start and stop
  - range(0, 7) implies range(0, 7, 1) and will return => **0, 1, 2, 3, 4, 5, 6**
- If three arguments then it is assumed to be start, stop and step
  - range(0, 10, 2) is a full and will return => **0, 2, 4, 6, 8**

For loops

- These are used when dealing with iterables

```
for item in items:
    'do something with item'
```

```
fruits = ['apple', 'pineapple', 'banana', 'guava']
for fruit in fruits:
    print(fruit)
```

For loops showing indexes - The enumerate() function

- Sometimes you want to see not just the items in a collection but ther indexes too.
- Remember when you were asked to enumerate in an exam or test in school? it simply meant number your answers.
- The same principle can be used in python. Another reason python is mostly English language.

```
for i, item in enumerate(items):
    print(i, item)
```

```
days = ['Sunday', 'Monday', 'Tuesday', 'Wednesday', 'Thursday', 'Friday', 'Saturday']
for day in days:
    print(day)

for i, day in enumerate(days):
    print(i)
```

```
    #print(f"This is day {i}") or
    #print(f"This is day{(i+1)}")

for i, day in enumerate(days):
    print(i, day)
    #print(f"{day} is listed as number {(i+1)} in the week")
```

For loops with dictionaries (key, value pairs) - The .items() method

- Remember dictionaries are not indexed by numbers counting from 0
- Dictionaries are indexed by keys
- example:

```
details  = {'name': 'Trevor Donald', 'location': 'Eldorado', 'age': 32}

print(details['name'])
print(details['age'])

for info in details:
    print(info)

for key, value in details.items():
    print(key)
    #print(value)
    #print(f"His {key} is {value}")
```

Review of objectives

- Understand loops
- Understand loop control with the Boolean True
- Understand infinite loops
- Understand iterables
- Understanding increments
- Mixing Logic and Loops
- Introducing new functions today
  - range()
  - len()
  - enumerate()
- Introducing a new method for dictionaries
  - dict.items()

Gist of the day

- Get the pdf version
- Get the python gist
- Jupyter Notebook will be uploaded

Q&A

Thanks for listening!