Sharif University of Technology
Department of Computer Engineering

Fundamentals of Programming

Python Language

Arman Malekzadeh
PhD Candidate in Artificial Intelligence

Table of contents

Lists (cont.)

Lists: Indexing

- `students = ['ali', 'mark', 'miley', 'sarah', 'bill']`
- `students[2]` # 'miley'
- `students[2:4]` # ['miley', 'sarah']
- `students[2:]` # ['miley', 'sarah', 'bill']
- `students[:2]` # ['ali', 'mark']
- `students[-1]` # 'bill'
- `students[-2]` # 'sarah'
- `students[-2:]` # ['sarah', 'bill']

Lists: Most Common Operations

- `students = ['ali', 'mark', 'miley', 'sarah', 'bill']`
- `len(students)` # 5
- `students.append('jane')` # [‘ali’, ‘mark’, ‘miley’, ‘sarah’, ‘bill’, ‘jane’]
- `students.insert(2, 'john')` # [‘ali’, ‘mark’, ‘john’, ‘miley’, ‘sarah’, ‘bill’, ‘jane’]
- `students.remove('miley')` # [‘ali’, ‘mark’, ‘john’, ‘sarah’, ‘bill’, ‘jane’]
- `students.pop()` # [‘ali’, ‘mark’, ‘john’, ‘sarah’, ‘bill’]
- `students.pop(2)` # [‘ali’, ‘mark’, ‘sarah’, ‘bill’]

Lists: Most Common Operations

- `students.index('mark')` # 1
- `students.count('mark')` # 1
- `students.sort()` # [‘ali’, ‘bill’, ‘mark’, ‘sarah’]
- `students.reverse()` # [‘sarah’, ‘mark’, ‘bill’, ‘ali’]
- `students.clear()` # []
- `tmp = students.copy()` # tmp = [‘sarah’, ‘mark’, ‘bill’, ‘ali’]
- Note: `tmp = students` will not copy `students`! It will just set `tmp` to point to `students`.
- Extend: `students.extend(['jane', 'john'])` # [‘sarah’, ‘mark’, ‘bill’, ‘ali’, ‘jane’, ‘john’]

Conditional Statements

Conditional Statements: `if` (cont.)

- Short-hand `if`

```
if a > b: print("a is greater than b")
```

- Nested `if` statements

```
if a > 2:
    print("a is greater than 2")
    if a > 3:
        print("a is also greater than 3")
    else:
        print("a is not greater than 3")
elif a == 2:
    print("a is equal to 2")
else:
    print("a is less than 2")
```

Loops

Subsection 1

`for` Loop

Loops: `for`

- `for` loop is used to iterate over a sequence (list, tuple, string) or other iterable objects.

```
for i in range(5):
    print(i)
```

- This for loop will print out the numbers 0 through 4. Let's break it down:
  - `for i in range(5):` This line is setting up our for loop. The `range(5)` function generates a sequence of numbers from 0 up to (but not including) 5. The variable `i` is our loop variable, which takes on each value in the sequence one at a time.
  - `print(i)` This line is the body of our loop, which is executed once for each value of `i`. It simply prints out the current value of `i`.

Loops: `for`

```
words = ["apple", "banana", "cherry"]
word_lengths = {word: len(word) for word in words if "a" in word}
```

- This code does several things:
  - It declares a list `words` containing three string elements: `"apple"`, `"banana"`, and `"cherry"`.
  - Then it uses a for loop to create a dictionary `word_lengths`. This dictionary will have the words from the `words` list as keys and their lengths as values.
  - However, not all words are processed. The `if "a" in word` condition filters the words and only those containing the letter “a" are included in the dictionary.

Loops: `for`

```
words = ["apple", "banana", "cherry"]
word_lengths = {word: len(word) for word in words if "a" in word}
```

- So, after processing this for loop, the `word_lengths` dictionary will look like this:

```
{'apple': 5, 'banana': 6}
```

As you can see, 'cherry' is not included because it doesn't contain an 'a'.

Loops: `for`

```
numbers = [1, 2, 3, 4, 5, 6, 7, 8, 9]
even_sum = 0
odd_product = 1

for num in numbers:
    if num % 2 == 0:
        even_sum += num
    else:
        odd_product *= num

print('Sum of even numbers:', even_sum)
print('Product of odd numbers:', odd_product)
```

Loops: `for`

In this script:

1. We first define a list `numbers` containing the integers from 1 to 9.
2. We also define two variables: `even_sum` (initialized to zero) which will hold the sum of the even numbers in the list and `odd_product` (initialized to one) which will hold the product of the odd numbers.
3. We then iterate over each number `num` in the list using a for loop.

Loops: `for`

In this script:

4. Inside this loop, we use an if statement to check whether `num` is divisible by two (i.e., whether it's even). If it is even (`num % 2 == 0`), we add it to `even_sum`.

5. If the number is not divisible by two (i.e., it's odd), we multiply it with `odd_product`.

6. After the loop has processed every number in the list, we print out both 'Sum of even numbers' and 'Product of odd numbers'.

Subsection 2

`while` Loop

Loops: `while`

- The `while` loop is used to repeatedly execute a block of code as long as a certain condition is true.

```
i = 0
while i < 10:
    print(i)
    i += 1
```

- This code will print the numbers 0 through 9, because the while loop will continue to execute as long as the condition `i < 10` is true.
- In each iteration of the loop, the value of `i` is printed and then incremented by 1.

Loops: `while`

```
i = 0
numbers = []

while i < 10:
    numbers.append(i)

    if i % 2 == 0:
        print(f"Adding {i} to the list.")
        print("This number is even.")
    else:
        print(f"Adding {i} to the list.")
        print("This number is odd.")

    print("Numbers now: ", numbers)
    i += 1

print("The final list: ", numbers)
```

Loops: `while`

In this script:

1. In this script, we first initialize a variable `i` to 0 and an empty list `numbers`.
2. Then we enter into a while loop which will continue running as long as `i` is less than 10.
3. Inside this loop, we first append the current value of `i` to our list `numbers`.

Loops: `while`

In this script:

4. Then we have an if statement that checks if the current value of `i` is even (this is what the condition `i % 2 == 0` does - it checks for divisibility by 2).

    - If it is even, it prints out a message stating that this number has been added to the list and that it's an even number. If not, it prints out similar messages but notes that the number is odd.

5. After this `if` statement, regardless of whether the current number was even or odd, it prints out the current state of our list `numbers`.

Loops: `while`

In this script:

6. Then we increment `i` by one (`i += 1`) before going back to check our while condition again.

7. Once `i` reaches 10 and our while condition fails, we exit out of the loop and print out our final list.

Code

All the code for this session is available Here:

References

References I

[1] B Downey, A. (2015). Think Python: How to Think Like a Computer Scientist-2nd Edition.

[2] Deitel, H. M., & Deitel, P. J. (2004). C: How to program. Pearson Educacion.

Sharif University of Technology
Department of Computer Engineering

Arman Malekzadeh

Fundamentals of Programming

Python Language