COMP10001

2023 Sem 1

Foundations of Computing

Week 4

Tutorial/Workshop ○ ○ ○ ○

Today's Tutorial

1. Methods vs functions
2. Lists and tuples
3. For and while loops

Methods

- run pre-defined code
- called with brackets that may have arguments
- are attached to an object
- we will not write these

```
e.g. "Hello".lower()
```

Functions

- run pre-defined code
- called with brackets that may have arguments
- called anywhere
- we will write these

```
e.g. print("Hello")
```

Exercise 1

Exercise 1 answer

1. Evaluate the following method calls given the assignment `s = "Computing is FUN!"` Think about the input and output of each method. You're not expected to know all methods for all types: if you haven't seen some of these before, your best guess based on the name will probably be right!

   (a) `s.isupper()`
   A: *False*

   (b) `s.upper()`
   A: *'COMPUTING IS FUN!'*

   (c) `s.endswith("FUN!")`
   A: *True*

   (d) `s.count('i')`
   A: *2*

   (e) `s.strip('!')`
   A: *'Computing is FUN'*

   (f) `s.replace('i', '!')`
   A: *'Comput!ng !s FUN!')*

Lists

- no size limit
- use when you are changing values
- can add/remove values

Initialise:

```
var = list("hello")
var = ["hi", 5, 3.2]
```

Tuples

- no size limit
- use when you don't change values
- cannot add/remove values
- can concatenate

Initialise:

```
var = tuple("hello)
var = ("hi", 5, 3.2)
```

Adding to lists

```
fruits = ['apple', 'banana', 'cherry']
animals = ['cat', 'dog']

fruits.append("orange")
 fruits = ['apple', 'banana', 'cherry', 'orange']
fruits.insert(1, "orange")
 fruits = ['apple', 'orange', 'banana', 'cherry']
fruits.extend(animals)
 fruits = ['apple', 'banana', 'cherry', 'cat', 'dog']
```

Removing from lists

```
fruits = ['apple', 'banana', 'cherry']


fruits.pop(2)
 fruits = ['apple', 'banana']
fruits.remove('apple')
 fruits = ['banana', 'cherry']
fruits.clear()
 fruits = []
```

Exercise 2

Exercise 2 answer

2. Evaluate the following given the assignment `lst = [2, ("green", "eggs", "ham"), False]`. Assume the list is reset after each part.

(a) `lst[2]`

A: *`False`*

(b) `lst[1][-2]`

A: *`"eggs"`*

(c) `lst[1][-2][:3]`

A: *`"egg"`*

(d) `lst.append(5); print(lst)`

A: *`[2, ("green", "eggs", "ham"), False, 5]`*

(e) `lst.pop(2); print(lst)`

A: *`False`* *(this is the value removed when .pop() is run)*
`[2, ("green", "eggs", "ham")]`

(f) `lst.reverse(); print(lst)`

A: *`[False, ('green', 'eggs', 'ham'), 2]`*

Iteration

Repeatedly executing a section of code

Makes code **cleaner**

Avoids "**hard-coding**"

e.g. can do something for the length of input, rather than coding for each possible length

for loops

```
for <loop variable> in <sequence>:
    *do something*

loop variable changes each time loop repeated

e.g.
for word in sentence:
    print(word)


for i in range(len(input)):
    print(input[i])
```

while loops

```
while <condition>:
    *do something*
```

keeps going while the condition is true

e.g.

```
i = 0
while i < 6:
    print("*")
    i++
```

```
i = 0
while i < 6:
    print("*")
```

Converting loops

It is **always** possible to convert for loop to while loops

Generally can convert while to for except if it waiting for user input

Challenging, common **MST** question!

Converting loops

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

>
0
1
2
3
4
```

```
count = 0
while count < 5:
    print(count)
    count = count + 1

>
0
1
2
3
4
```

Exercises 3&4

Exercise 3 answer

(a)
```
i = 2
while i < 8:
    print(f"The square of {i} is {i * i}")
    i = i + 2
```

A:
```
The square of 2 is 4
The square of 4 is 16
The square of 6 is 36
```

(b)
```
for ingredient in ("corn", "pear", "chilli", "fish"):
    if ingredient.startswith('c'):
        print(ingredient, "is delicious!")
    else:
        print(ingredient, "is not!")
```

A:
```
corn is delicious!
pear is not!
chilli is delicious!
fish is not!
```

Exercise 3 answer

(c)
```
i = 0
colours = ("pink", "red", "blue", "gold", "red")
while i < len(colours):
    if colours[i] == "red":
        print("Found red at index", i)
    i += 1
```

A:
```
Found red at index 1
Found red at index 4
```

(d)
```
MIN_WORD_LEN = 5
long_words = 0
text = "There once lived a princess"
for word in text.split():
    if len(word) >= MIN_WORD_LEN:
        print(word, "is too long!")
        long_words += 1
print(long_words, "words were too long")
```

A:
```
There is too long!
lived is too long!
princess is too long!
3 words were too long
```

Exercise 4 answer

(a)
```
i = 2
while i < 8:
    print(f"The square of {i} is {i * i}")
    i = i + 2
```

(a) **A:**

```
for i in range(2, 8, 2):
    print(f"The square of {i} is {i*i}")
```

Exercise 4 answer

(b)
```
for ingredient in ("corn", "pear", "chilli", "fish"):
    if ingredient.startswith('c'):
        print(ingredient, "is delicious!")
    else:
        print(ingredient, "is not!")
```

(b) A:
```
ingredients = ("corn", "pear", "chilli", "fish")
i = 0
while i < len(ingredients):
    ingredient = ingredients[i]
    if ingredient.startswith('c'):
        print(ingredient, "is delicious!")
    else:
        print(ingredient, "is not!")
    i += 1
```

WORKSHOP

Grok, problems from sheet, ask me questions :)

See you next week!