SKILLPILLS

Skill Pill: Intro to Programming

[Dictionaries], Strings, Control Structures

Jeremie Gillet

OIST

Outline

SKILLPILLS

- Dictionaries (in 5 minutes)
- Strings
- Control Structures

Dictionaries

```
  index      0   1   2   3   4   5   6
my_list = [1, 2, 3, 4, 5, 6, 7]

   keys     “hi”  (1,3)  range(4)
   values    1     2       3
```

```
my_dic = {“hi”:1, range(4):3, (1,3):2}
```

- A dictionary is an iterable that can contain anything
- Dictionaries are indexed with objects called keys

Manipulating elements

SKILLPILLS

- Retrieving an element
  - `val = my_dic[key]`
    - The key must be in the dictionary
- Adding an element
  - `my_dic[key] = val`

Dictionary methods

SKILLPILLS

- `len, in, max, sum, sorted # operates on keys`
- `d.get(key, default) # gives you back default if key isn't in d`
- `d.pop(key) # returns d[key] and deletes the pair from d`
- `d.keys() # returns the list of keys`
- `d.values() # return the list of values`
- `d.items() # returns a list of (key, value) tuples`
- `d.update(d1) # adds the values of d1 to d`

For loops

SKILLPILLS

- ```
  for key in d:
      print(key, d[key])
  ```
- ```
  d = { key:d[key] for key in d }
  ```
- ```
  d = { key: 10* d[key] for key in d
                        if type(d[key]) = int }
  ```


  ```

Exercise

dictionaries.py

Time: 0 minutes

Strings

- Strings are a way to store text
- Delimited by ‘ (single quote) or “ (double quote)
  - `x = “hello”`
  - `x = ‘this works’`
  - `x = “this ‘works’ too”`
- Strings are immutable iterators

Strings acting like lists

SKILLPILLS

- `“hello”[0] # returns “h”`
- `“hello”[1:3] # returns “el”`
- `“help” in “hello” # returns False`
- `“hi” + “ ” + “mom” # returns “hi mom”`
- `len(“hello”) # returns 5`
- `min(“hello”) # returns “e”`
- `sorted(“hello”) # returns ['e', 'h', 'l', 'l', ‘o’]`
- However, lists cannot be mutated (cannot use append, insert, del…)

Special characters

SKILLPILLS

- Need to be escaped with a `\` (backslash)
  - Tabulation: `\t`
  - New line: `\n`
  - Quotes: `\"`, `\'`
  - Backslash character: `\\`

String methods

SKILLPILLS

- `s.upper(), s.lower() # returns string with upper/lower characters`
- `s.strip() # returns string without space characters left and right`
- `s.find(s1) # if s1 is inside s, returns the index of s where s1 starts, otherwise -1`
- `s.split(sep) # splits s into chucks separated by sep`
- `sep.join([s1, s2, …]) # joins the strings with separator sep`
- `s.replace(old, new) # replaces occurrences of old by new`
- `“1".zfill(3) # returns “001”`

Format

- "I like {} and {}." .format("apples", “oranges")
  # 'I like apples and oranges.’
- "I like {1} and {0}." .format("apples", "oranges")
  # ’I like oranges and apples.’
- "I like {1}, {0} and {0}." .format("apples", "oranges") # ’I like oranges, apples and apples.’
- “An apple is {price} yen”.format(price=5000)
  # 'An apple is 5000 yen’
- “Pi is {}”.format(math.pi) # 'Pi is 3.141592653589793’
- “Pi is {:f}”.format(math.pi) # 'Pi is 3.141593'
- “Pi is {:07.3}”.format(math.pi) # 'Pi is 0003.14’

Exercise

lists.py

Time: 30 minutes

Control Structures

SKILLPILLS

- We need to be able to make decisions depending on context and repeat actions
- If, else, elif
- While loops
- For loops
- Bonus: list comprehensions

If condition

SKILLPILLS

4 space indentation

Testing condition

Alternative condition

default action

```
age = 17

if age < 16:
    print(“Drink soda”)
elif age < 18:
    print(“Have a beer”)
else:
    print(“Peket it is!”)
```

- We can have many elif
- elif and else are optional
- If conditions can be nested

While loops

```
a=0
while a < 5:
    print(a)
    a += 1
print("Done!")
```

- The indented instructions keep running while the condition is true
- Beware infinite loops (ctrl + c or red square will break the computation)

For Loops

SKILLPILLS

```
for i in range(1,1000):
    print(i)
    if i==256:
        print("Yay!")
```

- Loops through the elements of a list (or any iterable)
- Prefer a for loop to a while loop whenever you can!

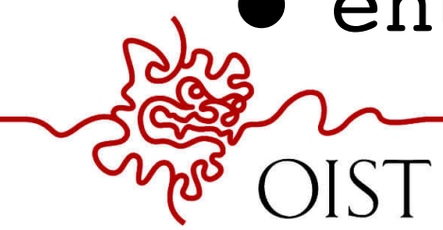
Unpacking within a loop

SKILLPILLS

```
for x in [(1,2), (3,5)]:
    print(x[0], x[1])


for i, j in [(1,2), (3,5)]:
    print(i, j)
```

- If the elements in the iterable are in a tuple or a list, they can be unpacked directly
- Related functions:
  - `zip([4, 9], “hi") # [(4, “h”), (9, “i”)]`
  - `enumerate(“hi”) # [(0, “h”), (1, “i”)]`



Aggregating

SKILLPILLS

```
mums = [2, 49, -7, 0, 14]

sum_nums = 0
for n in nums:
    sum_nums += n
print(sum_nums)

double_nums = []
for n in nums:
    double_nums.append(2*n)
print(double_nums)
```

List Comprehensions

SKILLPILLS

- For loops can be create a new list from the elements of another
  - `[i*i for i in range(10)]`
- You can filter the list
  - `[i*i for i in range(10) if i!=5]`
- You can nest loops
  - ```
    [(i, j) for i in range(10)
            for j in range(10,20)]
    ```


    ```

Exercise

loops.py

Time: until the end