KUAS
KYOTO UNIVERSITY OF ADVANCED SCIENCE
京都発世界人財

京都先端科学大学

Information Processing 1 13

files

projects: number games, function plotter
file sorter

Ian Piumarta

Faculty of Engineering, KUAS

last week: tuples

a tuple is a sequence of values, similar to a list

```
>>> ()
()
>>> "hello",
("hello",)
>>> a = (1, 2, 3)
>>> a
(1, 2, 3)
>>> len(a)
3
>>> for x in a: print(x, end=' ')
1 2 3
```

list operators work on tuples

```
>>> a = "one", "two", "three", "four", "five"
>>> a[1:-1]
("two", "three", "four")
>>> ('a', 'b') + ('c', 'd')
('a', 'b', 'c', 'd')
>>> ('a', 'b') < ('c', 'd')
True
>>> 'a' in ('a', 'b', 'c')
True
```

tuples as muliple values

```
>>> a, b = (1, 2)
>>> a
1
>>> b
2
>>> def minsec(secs):
        return secs // 60, secs % 60
>>> mins, secs = minsec(130)                    # divmod(130, 60)
>>> mins
2
>>> secs
10
```

tuples as dictionary key-value pairs

```
>>> d = { "one": "ichi", "two": "ni", "three": "san" }
>>> for kv in d.items(): print(kv)
("one" "ichi")
("two" "ni")
("three" "san")


>>> for key, val in d.items(): print(key, "->", val)
one -> ichi
two -> ni
three -> san
```

quiz

quiz

quiz

1. A *tuple* such as `(1, 2, 3)` is an ordered collection of elements, similar to a list. What is the *minimum* number of elements that a tuple can contain?

   1. Zero.
   2. One.
   3. Two.

2. What is the correct syntax to create a tuple the single element `42`?

   1. (42)
   2. 42,
   3. tuple(42)

quiz

3. If `a` is a three-element tuple (for example, `a=(1, 2, 3)`) then what is the correct way to change the second element to 22?

1. `a[1] = 22`
2. `a[1:2] = (22,)`
3. `a = a[:1]+(22,)+a[2:]`

4. If `a=(1, 2, 3)` then what is the value of `a` after executing the assignment `a, b, c = a`?

1. `(1, 2, 3)`
2. `1`
3. the assignment will fail (only one value on the right of the =).

quiz

5. The method `dict.items()` returns what result?
   1. A sequence of tuples, each one containing a key and corresponding value from the dictionary.
   2. An integer giving the number items (entries) in the dictionary.
   3. A tuple containing two elements: a list of the keys in the dictionary, and a list of their corresponding values.

6. Lists *cannot* be used as dictionary keys, but tuples *can*. Why?
   1. Lists can change their length but tuples cannot change their length even if their elements change.
   2. Tuples are immutable and their elements cannot be changed at all.
   3. Because the assignment in '`for key, val in dict.items():`' works for tuples but not for lists.

quiz

7. What is the value of `sorted((1, 3, 4, 2))`?
   1. An error because tuples are immutable and cannot be sorted.
   2. `(1, 2, 3, 4)`
   3. `[1, 2, 3, 4]`

8. What is the value of `(1,)*10`?
   1. `(10,)`
   2. `(1, 1, 1, 1, 1, 1, 1, 1, 1, 1)`
   3. An error because multiplying a tuple by an integer is not allowed.

this week: files

up to now our data structures have been *volatile*

- when the program terminates, the data is lost
- strings, lists, dictionaries, tuples

files provide *persistent* storage

- when the program terminates, data stored in a file is retained
- documents, music, videos, applications

opening and reading data from a file

```
fin = open("words.txt")

fin.read()          #=> next character from words.txt
fin.read(n)         #=> next n characters from words.txt

fin.readline()      #=> next line of text from words.txt

for line in fin:    #=> each line of text in words.txt
    print(line)
```

opening and writing data to a file

```
>>> fout = open("data.txt", 'w')    # 'w' = open for writing

>>> fout.write("hello, world\n")       # write data
13                                     # characters written
>>> fout.write("that's all, folks\n")
18
fout.close()  # send all data to disk and close the file
```

closing a file makes sure all data is written to the disk

once closed, `fout` cannot be read or written until it is opened again

question: what happens if you `open("data.txt", 'w')` again?

data written to a file must be a string

```
>>> x = 52
>>> fout.write(x)
TypeError: write() argument must be str, not int
>>> fout.write(str(x))
2
>>> fout.close()
```

formatting data for printing or for writing to a file

use the % operator to embed data from a tuple within it

```
>>> n, d = 3, 4
>>> "the %s %d / %d is equal to %g" % ("fraction", n, d, n/d)
"the fraction 3 / 4 is equal to 0.75"
```

- `%d` embed an integer in decimal
- `%x` embed an integer in hexadecimal
- `%s` embed a string as itself
- `%e` embed a float in engineering (exponent) notation
- `%f` embed a float with a fixed number of decimal places
- `%g` embed a float with a flexible number of decimal places

os, paths, and the current working directory

files and directories (folders) are organised into a tree structure

the module os provides functions for working with files and directories

```
>>> import os
>>> os.getcwd()                # get current working directory
'/Users/piumarta/Documents' # the path to the CWD
```

absolute paths start with '/', others are *relative* to the CWD

```
>>> os.path.abspath("data.txt")
'/Users/piumarta/Documents/data.txt'
>>> os.path.abspath("../temp.txt")  # .. is parent directory
'/Users/piumarta/tmp/data.txt'
```

testing files and direcories

```
>>> os.path.exists("data.txt")
True
>>> os.path.exists("no-such-file.txt")
False

>>> os.path.isdir("data.txt")
False
>>> os.path.isfile("data.txt")
True
>>> os.path.isdir("/Users/piumarta")
True
```

listing the contents of a directory

`os.listdir(`*dir*`)` returns a list of file and directory names within *dir*

```
>>> os.listdir("/Users/piumarta/Sync/KUAS/Class/IP1-ATPP/src")
['hello.py', 'function-plot.py', 'function-plot-auto.py',
'test.py', 'number-guess-mine.py', 'words.txt',
'database.db', 'number-guess-yours.py', 'data.txt']
```

`os.path.join(dir, name)` makes a path name from `dir` and `name`

```
>>> os.path.join(os.getcwd(), "data.txt")
'/Users/piumarta/Documents/data.txt'
```

exercise: print all file names in or below a given directory

step 1: write a function `walk(dirname)` that prints a list of file and directory names inside `dirname`

step 2: add a loop that prints each name on its own line

step 3: instead of names, print paths made by joining the `dirname` with each `name`

step 4: add an `if` statement that prints
- `"directory"` if the path refers to a directory
- the path itself if it refers to a file

step 5: instead of printing `"directory"`, use `walk(path)` to print the contents of subdirectories recursively

exercise: print all file names in or below a given directory

```
def walk(dirname):
    for name in os.listdir(dirname):
        path = os.path.join(dirname, name)
        if os.path.isdir(path):
            walk(path)
        else:
            print(path)

walk("/Users/piumarta/Sync/KUAS/Class/IP1-ATPP/src")
```

using exceptions to handle errors

```
def dangerous(name):
    names = os.listdir(name)
    print("continuing...")
    return names

print(dangerous("."))

print(dangerous("not-there"))
```

using exceptions to handle errors

```
def dangerous(name):
    names = []
    try:
        names = os.listdir(name)
    except:
        print("something bad happened")  # generic message
    print("continuing...")
    return names

print(dangerous("not-there"))
```

using exceptions to handle errors

```
def dangerous(name):
    names = []
    try:
        names = os.listdir(name)
    except Exception as e:
        print(e)              # specific message
    print("continuing...")
    return names
```

using exceptions to handle errors

```
def walk(dirname):
    try:
        for name in os.listdir(dirname):
            path = os.path.join(dirname, name)
            if os.path.isdir(path):
                walk(path)
            else:
                print(path)
    except Exception as e:
        print(e)

walk("/Users/piumarta/Sync/KUAS/Class/IP1-ATPP/src")
```

evaluating expressions in strings

the `eval(str)` function evaluates a string as a Python expression

```
>>> x = 21
>>> s = "x * 2"
>>> print(s)
"x * 2"
>>> print(eval(s))
42
```

this is most useful when the string comes from user input

```
>>> print(eval(input("expression: ")))
expression: x*3
63
```

end-of semester projects and evaluation

instead of exam: submit some completed projects

- submit 5 complete, tested, working projects
- you will earn the equivalent of 10 points per project

test your programs thoroughly!

each non-working project submitted *loses* you 5 points

- only submit a project that you know is working properly!

projects and points

last week	random words Markov analysis
this week	guess the computer's number guess your number function plotting find the largest files
future	database management graphical physics simulation sorting and searching

mini project (1): guess the computer's number

the computer picks a random number between 1 and 100
you guess the number, the computer tells you "too high" or "too low"
when you guess correctly the computer prints the number of guesses
you can choose whether to play again

`random.randint(low, high)` is a random integer in $[\text{low}, \text{high}]$

useful function: `getYesNo(prompt)`

- accept any prefix of "yes" or "no", ignoring case
- loop until a valid response is received

mini project (1): guess the computer's number

```
I have thought of a number between 1 and 100.
Try to guess the number.
Your guess: 50
Too high.
Your guess: 25
Too low.
...
Your guess: 26
You guessed the number in 7 attempts.
Play again (yes/no): y
I have thought of a number between 1 and 100.
Try to guess the number.
Your guess:
```

mini project (2): the computer guesses your number

you pick a random number between 1 and 100

the computer guesses your number, you tell it “high” or “low” or “correct”

when the computer guesses correctly it tells you the number of guesses

you can choose whether to play again

if you cheat then the computer should detect that and complain

binary search: you think of 34,

`lo`	`hi`	`(lo+hi)//2` *guess*	*response*	*update*
1	100	50	high	`hi` = 49
1	49	25	low	`lo` = 26
26	49	37	high	`hi` = 36
26	36	31	low	`lo` = 32
32	36	34	correct	

mini project (2): the computer guesses your number

```
Think of a number between 1 and 100.
I will try to guess it.
Each time I guess, tell me if I am HIGH, LOW, or CORRECT.
My guess is: 50
Am I HIGH, LOW, or CORRECT: h
My guess is: 25
Am I HIGH, LOW, or CORRECT: l
My guess is: 37
Am I HIGH, LOW, or CORRECT: h
My guess is: 31
Am I HIGH, LOW, or CORRECT: l
My guess is: 34
Am I HIGH, LOW, or CORRECT: c
I got it in 5 attempts.
Play again (yes/no): n
```

mini project (3): function plotting

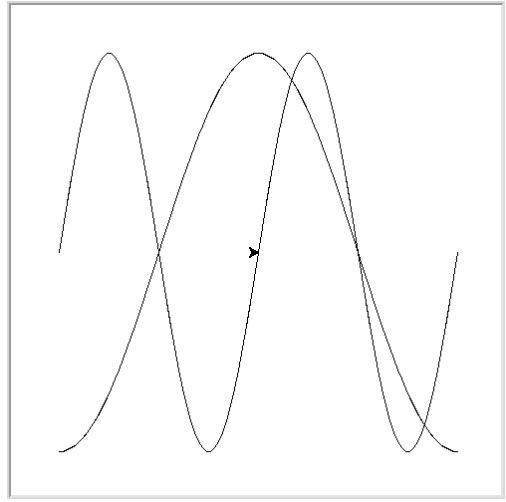
you can enter a function f in terms of x
you can set a minimum x value and a maximum x value
the computer plots your function $y = f(x)$ between $x_{\min}$ and $x_{\max}$

store function, $x_{\min}$, and $x_{\max}$ as strings
menu system to modify function, $x_{\min}$, and $x_{\max}$ as strings
use `eval(s)` to convert $x_{\min}$ and $x_{\max}$ to floats
use `eval(s)` to evaluate function for each x between $x_{\min}$ and $x_{\max}$
'`from math import *`' and '`from turtle import *`'
`setup(500, 500)` creates window, `clear()` clears it
`up()`, `down()`, `goto(x, y)` to plot the function

mini project (3): function plotting

```
1) change function: sin(x)
2) change minimum:  -2*pi
3) change maximum:  2*pi
4) plot function
5) exit

choice: 4

1) change function: sin(x)
2) change minimum:  -2*pi
3) change maximum:  2*pi
4) plot function
5) exit

choice: 1
new function: cos(x)

1) change function: cos(x)
2) change minimum:  -2*pi
3) change maximum:  2*pi
4) plot function
5) exit

choice: 2
new minimum: -pi
```

```
1) change function: cos(x)
2) change minimum:  -pi
3) change maximum:  2*pi
4) plot function
5) exit

choice: 3
new minimum: pi

1) change function: cos(x)
2) change minimum:  -pi
3) change maximum:  pi
4) plot function
5) exit

choice: 4

1) change function: cos(x)
2) change minimum:  -pi
3) change maximum:  pi
4) plot function
5) exit

choice: 5
```



mini project (4): list files in order of size

find (recursively) all files below a certain directory

for each file, find the size of the file

sort the results into order of increasing file size

print each file name preceded by its size

very useful for finding out where your disk space is being used!

`getfiles(dirname, files)` adds files under `dirname` to `files`

`getsizes(files)` makes list of tuples `(size, filename)`

a list of tuples can be sorted and the result printed

to write the above two functions you can use

- the function we wrote in class to find all files under a directory
- `os.path.getsize(path)` to find the size of the file at `path`

mini project (4): list files in order of size

for extra kudos, read the initial directory from the command line

```
import sys
print(sys.argv)
if (len(sys.argv) > 1):
  print("first argument is:", sys.argv[1])
```

projects

download project descriptions in this week's assignment

complete and then submit five finished (working) projects

if you have difficulty starting, download some **pseudo-code templates**

- look in General channel, Files section, Week 13 folder for:
  `ip1w13p01.py` (guess-computer-number)
  `ip1w13p02.py` (guess-user-number)
  `ip1w13p03.py` (function-plot)
  `ip1w13p04.py` (file-sizes)

10:30	10	last week: tuples
10:40	5	files and persistent storage
10:45	5	opening and reading a file
10:50	5	opening and writing a file
10:55	5	data written must be a string
11:00	5	formatting data for printing or writing
11:05	5	os paths and the current working directory
11:10	5	testing files and directories
11:15	5	listing the contents of a directory
11:20	20	exercise: print all file names below a directory
11:40	5	using exceptions to handle errors
11:45	0	evaluating expressions in strings
11:45	10	quiz
11:55	5	end-of-semester projects and evaluation
12:00	40	lunch
12:40	5	project 1: guess computer's number
12:45	5	project 2: guess user's number
12:50	5	project 3: function plotting
12:55	5	project 4: list files in order of size
13:00	70	projects
14:10	00	end