

Principles of Programming

Hugh Davis

ECS

The University of Southampton, UK

users.ecs.soton.ac.uk/hcd

The Principles

These are the things you need to have in pretty much any programming language;

- Assignment
- Input/Output
- Sequence
- Selection
- Iteration
- Modularity and Functions (Built on and user-defined)

Assignment and Stored State

```
turns = 5
```

```
guesses = 'aeiou'
```

The above are not philosophic statements! They are assignments.

turns is a piece of memory of a shape that expects to store integers.

This piece of memory has been told to store 5

The line

```
turns = turns + 1
```

Has the effect of adding 1 to 5 and assigning the result (6) back to turns

guesses is a piece of memory that can store random sized strings. The string aeiou has been stored in this memory

Input/Output

- If you cannot make inputs then the program will always keep doing the same thing!
- If you cannot get outputs then how will you know what happened!

Most languages have some kind of input for strings from the keyboard and output of strings in “typewriter” mode

```
guess = raw_input ('guess a letter: ')\nprint ('Hello World!')\nprint ('You took', x, 'goes')
```

Languages that support WYSIWIG will have more sophisticated “event” input and graphical output

Sequence

Each instruction will be executed in the order it is encountered

```
import random

import urllib

print ('time to play hangman')

animals =
urllib.urlopen('http://davidbau.com/data/animals').
read().split()

secret = random.choice(animals)
```

```
guesses = 'aeiou'
```

Selection

- The ability to choose which set of actions are taken next

```
if letter in guesses:
```

```
    print (letter),
```

```
else:
```

```
    print ('_'),
```

```
    missed += 1
```

and

```
if turns <5:print(' 0  ')
```

```
if turns <4:print('\_|\_/ ')
```

```
if turns <3:print('  | ')
```

```
if turns <2:print(' / \ ')
```

Iteration

- The ability to repeat a set of instructions

Construct like for, while, repeat are used

```
turns = 5
```

```
while turns > 0:
```

```
    print (turns)
```

```
    turns = turns - 1
```

and

```
for letter in secret:
```

```
    if letter in guesses:
```

```
        print (letter),
```

```
    else:
```

```
        print ('_'),
```

```
        missed += 1
```

Modularity

- The ability to take a bit of code that has a known behaviour and put it in a back box so that it can be used elsewhere in the code.

Strictly speaking Functions are chunks of code that take *parameters* (inputs) and *return* a result, and have no other effect on anything outside their black box (no *side-effects*)

Some languages allow functions to have side-effects

Some languages allow functions that do not return a result – and some languages call such functions *procedures*

Example Functions

```
def find_first_2_letter_word(xs):  
    for wd in xs:  
        if len(wd) == 2:  
            return wd  
    return ""
```

Used by

```
find_first_2_letter_word(["This", "is", "a", "dead", "parrot"])
```

- Built in (System Defined)

len is a system function which returns the length of a string as an integer

- User Defined

```
find_first_2_letter_word
```

is a function which you give a list of strings and it returns the first string that has two letters (or the empty string)