# ICS 31 Thursday 10 October 2013, 11 am

Things/nouns/objects/data structures

Actions/verbs/functions, methods, procedures, operators/control structures

Control structure for modularity/encapsulataion:

Programmer-defined functions

Syntax and semantics of function definitions/calls

Syntax:

Def FUNTIONNAME ( PARAMETERLIST ) -> RETURNTYPE :

“DOCSTRING, short description of what the function does ”

FUNCTIONBODY

Assert BOOLEABEXPRESSION

FUNCTIONNAME: A Python identifier, we choose

PARAMETERLIST: zero or more of these, separated by commas

PARAMETERNAME: TYPE

TYPE: A data type in Python (int, Restauraunt, [str])

FUNCTIONBODY is one or more statements, with the last one usually

Return EXPRESSION

ASSERTIONS are examples of calling the function with the result you expect. There are your tests.

Def n\_copies(n: int, s: str) ;> str:

“Return a string, n copies of the parameter s”

return n \*s

assert n\_copies(5,’X’) == ‘XXXXX’

assert n\_copies(0, ‘hello’) == ‘’

Semantics:

N\_copies(10,’?’) # a function CALL

1. The arguments (10 and ‘?’ are PASSED to the function:

In the definition, n becomes 10, x becomes “?”

The parameters, n and s, are the function’s INTERNAL names

For its inputs/arguments.

1. The body of the function is executed/evaluated (presumably using the parameters
2. The return statement sends its value back to the calling program.

Value of above expression is ‘??????????’

Print(‘Hello.’ N\_copies(3,’ok’) +’!!’, ‘Goodbye.’\_

Print (‘Hello.’, ‘okokok’ +’!!’, ‘Good-bye’)

Print (‘Hello.’, ‘okokok!!’, ‘Good-bye.’)

Hello. Okokok!! Good-bye.

More Control structures

Besides functions for modularity/encapsulation

Nested expressions: (a + b\_ \* len(c)/d)

Sequence of statements

Selection: Ask a yes/no (Boolean) question, do different things

Depending on the result. (if statements)

Repetition: Do a block of code more than once.

Most often: Do function F for each time in a list

(or each character in a string, …)

Other variations on repetition