• Questions
• Hello, Python, cont’d
• Python functions
  • Built in functions
  • User-defined functions

• Programming (CS tool) = CT + Coding
• High level languages: Python, etc.
• Python primitive elements
  Objects (type/value/id)
• Primitive elements can be combined
• Expressions evaluate to a value
• Assignment statements (are not expressions) associate name/object
• Variables (names) are expressions

>>> x = 0
>>> x = x + 1
>>> x = x + 2
>>> x

Syntax Error:
>>> x == 5
SyntaxError: invalid syntax

>>> 5 = x
SyntaxError: can't assign to literal

Bug Collection So Far
NameError:
>>> who
Traceback (most recent call last):
  File "<pyshell#11>", line 1, in <module>
    who
NameError: name 'who' is not defined

Bug Collection So Far
2^4 [where 2 ** 4 is intended]
Python Functions

A function names an operation.

Why functions?
Readability/Clarity
Reuse

Python has built in functions and user-defined functions

For example,
>>> abs(-8)
>>> y = 1000.001
>>> round(y)
>>> pow(2, 4)

User-defined functions associate a name with a user-defined operation/process.

x = 99

def add_one(x):
    y = x + 1
    return y

A function definition is not an expression and is not evaluated.

As with built in functions, function must be called:
>>> add_one(4)
5

A function call is an expression and is evaluated.

Computers do not solve problems - computers carry out solutions, specified by people, to problems.

Questions
Python functions, cont’d
• Defining v. Calling
• Parameters v. Arguments
• Local variables
• Memory model of function call
• Function design recipe
Parameters v. Arguments

```python
>>> def add_one(x):
    y = x + 1
    return y

>>> add_one(101)
x is assigned the value 101 when the function is called
```

Local Variables

```python
>>> def add_one(x):
    y = x + 1
    return y

y is a local variable
```

What is the output of this code?

```python
def calculate(w, x):
    a = x
    b = w + 1
    return a + b + 3

>>> calculate(1, 5)
>>> calculate(2, 0)
```

Functions can call functions:

```python
def calculate(w, x):
    a = x
    b = add_one(w)
    return a + b + 3

>>> calculate(1, 5)
>>> calculate(2, 7)
```

(9) What value is returned after the following code is entered into the Python shell?

```python
>>> def calculate(w, x):
    a = x
    b = w + 1
    return a + b + 3

>>> calculate(2, 7)
```

The Butte to Butte road race, held in Eugene each year on the 4th of July, starts with runners racing up and then down a hill. The uphill distance and downhill distance are known (1 and 2 km, respectively*), and the uphill time and downhill time are available after the racer has completed those sections of the course. Prizes are given for the fastest up hill time and the fastest downhill time. This year we are proposing a new prize for the fastest average speed for the entire uphill/downhill portion of the race.

Write a function, **up_down_time**, that will print out the average speed (in km/min) for the entire uphill/downhill portion of the race.
Step 0.a:

THINKING
what is the desired outcome?
what type of value is this?

THINKING
what is the desired outcome?
what type of value is this?

Step 0.b:

THINKING
what information are we starting with?
what type of value(s) is this?

THINKING
what information are we starting with?
what type of value(s) is this?

Step 1:

THINKING
what will be the function output?
what is input to the function?
parameters
local variables

THINKING
what will be the function output?
what is input to the function?
parameters
local variables

Step 1: examples*

up_down_time(2, 1)
1.0

up_down_time(4.5, 8)
<about .25>

up_down_time(2, 1)
1.0

up_down_time(4.5, 8)
<about .25>

*these can be used to test the function

Step 2 (and 3 and 4) Start writing a template for the function:

a) header
b) “pass”
c) return statement
d) type contract
e) examples
f) description
def up_down_time(uptime, downtime):
    
    (float, float) -> float

    Return the average speed (km/min) on the combined uphill and
downhill parts of a road race, given the distance (in km) of each
part of the race and the up and down times (args).

    >>> up_down_time(2, 1)
    1.0
    >>> up_down_time(4.5, 8)
    <about .25>
    
    pass  # code goes here
    return #avg_speed

All of your functions should look like this template!