Python Research Project

Mike Ripley
Fall 2004
ripleymj@jmu.edu

Question Set 1

1. What is the URL for this language?

http://www.python.org

2. What individual or what company developed the language?

The Centrum voor Wiskunde en Informatica, or National Research Institute for Mathematics and Computer Science in the Netherlands. Development is now handled by the Python Foundation.

3. What paradigm is it an example of?

Python is an object-oriented language.

4. What domain is it useful in?

Python is a full-featured scripting language, much like PHP and Perl.

5. Is the language compiled or interpreted?

Python is interpreted independently from the host platform.

Question Set 2

1. How is the syntax of your language described? (i.e. using BNF, with syntax diagrams, with examples, in words)

http://www.python.org/doc/2.3.4/ref/ref.html - The language is described in words, and illustrated with brief examples. Wikimedia also has a decent reference on Python, for example: http://en.wikibooks.org/wiki/Programming:Python_Input_and_output

2. With regard to variable names (not values), what characters are allowed?

Identifiers must begin with either a letter or an underscore character. Following that, an unlimited sequence of letters, numbers, or underscore characters may be used.

3. With regard to variable names (not values), what is the maximum length allowed and how many characters are significant?

Python seems to truncate variables at 255 characters when concatenating private names.

4. What is the assignment operator in your language?

Python uses the single equals, =, for assignment.

5. Does your language have reserved words and/or key words?

Python has 29 reserved words.

and	del	for	is	raise
assert	elif	from	lambda	return
break	else	global	not	try
class	except	if	or	while
continue	exec	import	pass	yield
def	finally	in	print	

6. Show a typical assignment statement.

self.balance = self.balance + depositAmount

Note that Python uses the newline as a statement terminator, not a semi-colon. To span lines, use a backslash, \.

7. What types of selection statements are available in your language?

Python uses:

if/then

if/then/else

if/elif/else

8. What types of iterative statements are available in your language?
Python uses:
for x in boolean
for x in iterator
while boolean

9. Show a typical input statement in your language.
Python supports two input statements: `raw_input()` and `input()`. Raw_input() will read a string from the command line, while input() will attempt to parse the input on the command line as code. Using input(), an array can be entered directly into the program.

```
raw_input('What is your name?')
```

10. Show a typical output statement in your language.
Python uses `print()` with C `sprintf()` formatting.

```
>>> print 'The value of PI is approximately %5.3f.' % math.pi
The value of PI is approximately 3.142.
```

Question Set 3

1. Show a typical comment in your language
Comments begin with a '#'.
Example:
#Programmer: Mike Ripley

2. Where does a comment begin, end?
A comment begins at the beginning of the line, and continues to the end, unless the line joining character, '\'.

3. Where can it be placed?
In addition to leading a line, it can be indented, or follow code.

4. Can a single comment extend over more than one line?
Yes, use the line joining character.

5. Is your language case sensitive?
Yes, though there is some discussion about removing the restriction.

6. Does your language allow and/or require type declarations?
Like most scripting languages, type declarations are not required. A means to manually specify type is not immediately apparent.

7. What are the scalar data types, if any, in your language?
Integers, Floating Points, Complex. A boolean is considered an integer in python.

8. What are the structured data types, if any, in your language?
The line between "other" and "structured" is blurred by the Python rule that everything is handled as an object. See the list below for the complete listing of variable types.

9. Does your language have any other data types (i.e. pointers???)
None
NotImplemented
Ellipsis
Immutable sequences
    Strings
    Unicode
    Tuples
Mutable sequences
    Lists
Mappings
    Dictionaries

Callable types
    User-defined functions
    User-defined methods
    Generator functions
    Built-in functions
    Built-in methods
    Class types
    Classic Classes
    Class instances
Modules
Classes
Class instances
Files
Internal types
    Code objects
    Frame objects
    Traceback objects
    Slice objects
    Static method objects
    Class method objects

10. How are strings handled in your language? (look at our text for a description of the possible choices before answering this question).
Since strings are listed as immutable by the language reference guide, I'd imagine they're implemented as static length. Any change will allocate a new static block to hold the new string.

Question Set 4

1. Where did you find a compiler/interpreter?

The Python interpreter is included in the Debian GNU/Linux package management system. Windows users might want to try the interpreter offered at python.org, or the ActiveState ActivePython program. Both are available at no cost.

2. What was the name of the file you downloaded to install it?

python2.3_2.3.4-13_i386.deb

3. What are the steps you used to install it (i.e. provide the directions you actually followed to install it).

Issue the following command as the system superuser:

apt-get install python2.3

4. How do you compile/interpret a program? (i.e. What command do you issue?)

Commands may be entered interactively, read from a file, given on the command line, or piped in on STDIN. To run a pre-written script, enter:

python filename.py

5. What environment are you using to edit the code?

I am using the vi editor.

6. How do you run a program? (show the command)

python filename.py

7. Attach the source code for a short program that you have interpreted/compiled and run. The program does not have to achieve any meaningful task but must include statements illustrating 3 of the answers you provided to questions 6,7, 8,9, and 10 from question sets #2. It must include line by line comments identifying the type of statement AND the requirements for the statement. Circle the type of statements you are illustrating.

a. Assignment

b. Selection

c. Iteration

d. Input

e. Output

```
#Loop over a code block
#range() takes a starting integer and a max. integer
#equivalent to for(int ii=1;ii<3;ii++)
for ii in range(1,3):
     #Indented code is part of the for block
     #Read a string, using question as a prompt
     name = raw_input('What is your name? ')
     #Write Hello plus the previously read string
     print 'Hello ', name
```

8. Capture the output produced by your program (screen capture). Print it and attach it also.

Screenshot omitted.

Question Set 5

NOTE: Python refers to the traditional array as a list. There is also an array class that may be imported. The functionality of the actual array is a subset of that of a list. Arrays require type definitions, while lists have no such requirement. The answers below refer to the list data type.

1. What symbols are used to delimit array indexes (i.e. ( ), [ ], something else)?

   Python uses [ ] to denote list indexes.

2. Give an example of a one dimensional array declaration.

```
a = ['spam', 'eggs', 100, 1234]
```

3. Give an example of a two dimensional array declaration.

```
q = [2, 3]
p = [1, q, 4]
```

4. What types are legal for subscripts?

   Though Python are variables are loosely typed, variables used as indexes must have integer values stored in them. Below is an example of an integer vs. non-integer index.

```
>>> a = ['spam', 'eggs', 100, 1234]
>>> b = 1
>>> a[b]
'eggs'
>>> b = 'spam'
>>> a[b]
Traceback (most recent call last):
  File "<stdin>", line 1, in ?
TypeError: list indices must be integers
>>>
```

5. Can arrays be initialized when they have their storage allocated?

   Yes, see the example given for question 2.

6. Can you initialize the entire array at once or do you have to initialize it item by item?

   It appears that lists must be initialized item by item.

7. Are ragged or rectangular multidimensional arrays allowed, or both?

   Ragged multidimensional arrays are allowed. If the raggedness just so happened to work out to be a rectangle, that's acceptable as well. The syntax presented in question 3 would expand to a much larger list.

8. What kind of slices are allowed, if any?

   A subset of an array may be taken by using array[lowerbound:upperbound] notation, instead of array[index] notation. The Python documentation provides this advise for slicing lists (assume s[i,j]):

   > If *i* or *j* is negative, the index is relative to the end of the string: `len(s) + i` or `len(s) + j` is substituted. But note that `-0` is still `0`. The slice of *s* from *i* to *j* is defined as the sequence of items with index *k* such that *i* `<=` *k* `<` *j*. If *i* or *j* is greater than `len(s)`, use `len(s)`. If *i* is omitted, use `0`. If *j* is omitted, use `len(s)`. If *i* is greater than or equal to *j*, the slice is empty.

9. Are array sizes static or dynamic?

   Array sizes are dynamic. Arrays may even be inserted into themselves using code like this, which would insert list a at the beginning of list a:

   ```
   a[:0] = a
   ```

10. Does your language detect a reference to an out of bounds array subscript?

Yes, an out of bounds error looks like this:

```
>>> a[4]
Traceback (most recent call last):
  File "<stdin>", line 1, in ?
IndexError: list index out of range
>>>
```

Question Set 6

1. Attach the source code for a short program that you have interpreted/compiled and run. The program does not have to achieve any meaningful task but must include statements illustrating 3 additional answers from question set #2 (the answers to questions 6,7, 8,9, and 10 that you didn't already illustrate in question set #4). It must include line by line comments identifying the type of statement AND the requirements for the statement. Circle the type of statements you are illustrating.

    a. Assignment
    b. Selection
    c. Iteration
    d. Input
    e. Output

```
#Iterate over a code block
#range() takes a starting integer and a max. integer
#equivalent to for(int ii=0;ii<3;ii++)
for ii in range(0,3):
        #Indented code is part of the for block
        #Input a number as code, using string as a prompt
        #Assign the result to "number"
        number = input('Enter a number: ')
        #Select a case and output a message
        if(number == 1):
                print 'You entered a one'
        elif(number == 2):
                print 'You entered a two'
        else:
                print 'You entered another number'
```

2. Capture the output produced by your program (screen capture). Print it and attach it also.

    Screenshot omitted.

3. Does your language provide any exception handling mechanisms?

    Python uses try/catch blocks to trap errors.

4. If it does, provide a description of what happens on erroneous input. If it doesn't describe what happens when on erroneous input.

    The interpreter will trace error back to the offending line of code and try to match the error to a known exception. Since Python is loosely typed, data errors are difficult to raise. Problems can stem from EOF and the input() function which tries to evaluate input as code. Any error in the code will throw the appropriate exception.

5. Can the user define their own exception handlers?

    Yes.

6. What types of subprograms does your language have (i.e. functions, procedures), if any?
   Python uses functions that are declared with the “def” statement.

7. What is the parameter passing mechanism in your language?
   Parameters are passed by reference.

8. Does your programming language allow recursive calls?
   Yes, in two variations. Traditional Python interpreters stored Python function calls in the C frame stack. There are experimental versions of the interpreter that move Python function calls to the C heap, allowing much deeper recursion.

Question Set 7

1. What was the most difficult thing for you to master in your language?

Python presented no immediate difficulties. The syntax is clean and easy to get start with.

2. What feature of your language did you really like that you hadn't seen before in another language?

The input() function, which evaluated stdin as code, was quite amazing.

3. Describe a problem or type of problem that is easy to solve in your language that would be more difficult if not impossible to solve in another language (preferably one that we have looked at).

Programming is a difficult problem in Ada that is easily solved with Python. The “one-right-way-to-do-it” philosophy that guided Python's development makes the convoluted governmentness of Ada look like a ball of mud.

4. What is the language in which the problem described above would be difficult to solve?

Ada.

5. Do you think the entire class should learn this language the next time this course is taught?

The class should be exposed to some scripting language, be it Python, Perl, or PHP.

6. Give your reason(s) for your answer to the previous question.

The flexibility and loose-typing provide an experience that is not often seen in traditional compiled languages.

7. What did you learn from learning this language?

I learned that this may be a language worth pursuing at a later date. The widespread deployment of the Python interpreter as well as the GUI bindings could make this an interesting rapid-prototyping language.

8. If you are asked on the final to give a brief description of your language (1 or 2 sentences) what will you say?

Python is a scripting language developed in the Unix world to provide a coherent and consistent object oriented environment for any task.

9. Would you like to have the responses to everyone's project questions?

Only some, such as languages still in widespread use.

10. What question should have been part of the project questions that wasn't included?

    What was the inspiration for development?
    How many revisions has the language standard seen?
    Is the language still true to it's original purpose?
    Did the language supersede any previous languages? Has it been superseded?
    Has the language evolved to meet new challenges while retaining it's original intent?
    Is the language still useful?