Review Session

March 12, 2017

Topics covered

- Print vs Return
- Expression evaluation
- Conditionals
- Strings
- Functions
- Testing

Print vs Return

- It just prints out the value on the terminal
- It allows you to see the values of variables during execution.
- It can be used outside a function.
- Eg.

```
def p_simple_interest(p, n, r):
        print p * n * r / 100.0
```

- It returns a value to the expression where it is called.
- It is used in program to pass results of complex operations.
- It must be used in the function definition.
- Eg.

```
def r_simple_interest(p, n, r):
        return p * n * r / 100.0
```

Print vs Return demo

Expression evaluation

- While evaluating expressions, we need to consider 2 factors:
  - Precedence
  - Associativity
- Precedence helps us in deciding the order of evaluating the expression **when there are different operators**.
- Associativity tells us which part of an expression should be evaluated first **if the operators are the same**.

Precedence and associativity

Precedence:

- Exponentiation: **
- Unary operators: + –
- Binary arithmetic: * / %
- Binary arithmetic: + –
- Comparisons: < > <= >=
- Equality relations: == !=
- Logical not
- Logical and
- Logical or

Associativity:

- Exponentiation: Right - to - left
- Everything else: Left - to - right

Precedence and associativity

Example 1:

```
>> 2 + 3 * 5 - 8 / 2

# 2 + 15 - 4

# 17 - 4

# 13 -> answer!
```

Example 2:

```
>> 3 / 5 + 2 ** 3 * 3

# 3 / 5 + 8 * 3

# 0 + 24

# 24 -> answer!
```

Evaluate the following expression:

2 ** 3 ** 2 ** 1

A) 64

B) 32

C) 512

D) I have a different answer

Expression: 2 ** 3 ** 2 ** 1

-> 2 ** 3 ** 2

-> 2 ** 9

-> 512

Conditionals

Conditionals

- It allows you to execute the statements that satisfy a certain criteria.

Format:

```
if <expression-1>:
        <action-1>
elif <expression-2>:
        <action-2>
...
else:
        <action-n>
```

Example: Conditionals

```
if temperature < 30:
        print “It’s freezing”
elif temperature >=30 and temperature < 60:
        print “It’s cold”
elif temperature >= 60 and temperature < 70:
        print “It’s warm”
else:
        print “It’s hot”
```

Strings

String operations

- Slicing
- Count
- Index
- Find
- Strip

Functions

Functions

FORMAT:

```
def <function-name>(<parameters>):
        ‘ ’ ’  <explanation>
        Precondition: <preconditions> ‘ ’ ’
        <action-1>
        <action-2>
        return <something>
```

Function definition v/s function call

```
def rectangle_perimeter(length, width):
    ‘ ’ ’ Calculates perimeter of a rectangle
    Precondition: length and width are floating
    point numbers greater than 0
    ‘ ’ ‘

    perimeter = 2 * ( length + width)
    return perimeter
```

```
>> rectangle_perimeter(4.0, 5.0)

18.0
```

Testing

Testing

- Develop a concrete understanding of the function you want to test.
- Understand the output of the function on a given input.
- Create inputs that can represent a set of inputs for the given function.
-

Example: Testing

$$X_1 = (-b + \sqrt{D})/2a$$
$$X_2 = (-b - \sqrt{D})/2a$$
where,
$$D = b^2 - 4ac$$

General Algorithm Design

Steps:

1. First understand what you want to do
2. Break it down into simple manageable steps
3. Understand the execution flow in the steps
4. Check whether your steps are correct
5. Write down Python code for it