Getting Started

Informatics I501

Week 01

Goal

- “That you would feel comfortable bringing computational thinking to bear on solving many of the problems you encounter during your studies, work and everyday life.”

Knowledge

- Declarative
  - Statement of a fact
  - “Chicago is north of Indianapolis”
- Imperative
  - How to do something
  - “To drive to Chicago, head up I65 North to I90 West, …”

  - Even deeper
  - How do you determine the “best” route between two locations

Algorithm

- “a sequence of simple steps, together with a flow of control that specifies when each step is executed.” p.2

- “a finite list of instructions that describe a computation that when executed on a set of inputs will proceed through a set of well-defined states and eventually produce an output.” ibid

Programming languages

- Low-level versus High-level
- General versus Domain specific
- Interpreted versus Compiled

Python

Python

- High-level, general purpose and interpreted
- Object oriented
- Dynamically typed

- Created in 1990 by Guido Von Rossum
  - Based on his experience developing ABC
  - Benevolent dictator – recently resigned
  - Python Software Foundation

Basic Elements of Python

- Programs (scripts)
  - Sequence of commands (or statements)
  - Objects, Expressions and Numerical Types
  - Variables and Assignment

Objects

- “core things that Python programs manipulate” (p. 9)
- Each object has a **type**
- A type is either **scalar** or **non-scalar**
  - Scalar types
    - **int** – integer, this is a “whole number”
    - **float** – real number – rational or irrational
    - **bool** – has a value of either True or False
    - **None** – special type
  - Non-Scalar types
    - Strings and things - Week 3
- Literals – 2, ‘IUPUI’

Expressions

- Sequence of objects and operations
- Evaluations to an object of some type
  - **Value** of the expression
- Follows typing rules
  - Expressions involving only integers have an integer result

Operators

- i + j; addition
- i – j; subtraction
- i * j; multiplication
- i ** j; i raised to the power of j; i^j

- If both are integers the result is an integer
- If one or both are floats the result is a float

Operators (division)

- i / j; division
  - Result is a float
- i // j; integer division
  - Result is the integer quotient
- i % j; modulo
  - Result is the integer remainder

Operators (logical)

- i == j; equality
- i != j; inequality
- i < j; less than
- i <= j; less than OR equal; i is ‘at most’ j
- i > j; greater than
- i >= j; greater than OR equal; i is ‘at least’ j

- Result is bool True or False

Variables and Assignment

- Variables assign a name to an object
- Python evaluates the **expression** to the right of the **assignment** symbol and then points the **variable** to the new object

```
pi = 3
radius = 11
area = pi * (radius**2)
radius = 14
```

Variable names matter

- “A rose by any other name would be just as sweet” (W. Shakespeare)
- “A variable with a bad name is very weak” (B. Green)

```
a = 3.14159
b = 11.2
c = a * (b**2)
```

```
pi = 3.14159
diameter = 11.2
area = pi * (diameter**2)
```

Some Coding Conventions

- Indent 4 spaces
  - Use spaces not tabs
- Constants (Literals): Use upper case
  - PI = 3.1415
  - MONTHS_PER_YEAR = 12
- Variables: Use lower case with words separated by underscores
  - area
  - mean_rainfall_per_month
- Python Style Guide (PEP 8)
- PSG Cheat Sheet

Reserved words (keywords)

```
and
class
elif
False
global
is
not
return
with
as
continue
else
finally
if
lambda
or
True
yield
Assert
def
else
for
import
nonlocal
pass
try
Break
del
except
from
in
None
raise
while
```

Comments

- Not interpreted or executed
- Start at the **#** and continue to the end of the line
- Can start and the beginning of a line or after a statement

```
#subtract area of square s from area of circle c
area_circle = PI*radius**2 #Area of the inscribed circle
area_square = side*side    #Area of the square
difference = area_circle - area_square
```

Comment Conventions

Comments		
Limit the line length of comments and docstrings to 72 characters.	Use complete sentences, starting with a capital letter.	Make sure to update comments if you change your code.

Block Comments		
Indent block comments to the same level as the code they describe.	Start each line with a # followed by a single space.	Separate paragraphs by a line containing a single #.

Inline Comments
Use inline comments sparingly.
Write inline comments on the same line as the statement they refer to.
Separate inline comments by two or more spaces from the statement.
Start inline comments with a # and a single space, like block comments.
Don't use them to explain the obvious.

Multiple assignment

- Can assign values to multiple variables at once

- `x, y = 2, 3`
- `x, y = y, x`

- `mon, tue, wed, thu, fri, sat, sun = 1, 2, 3, 4, 5, 6, 7`

Branching programs

- Straight-line programs and generally boring
- If some **conditional** is True then execute a block a statements
  - Conditional is a Boolean variable or an expression that evaluates to a Boolean value
  - Optionally execute a block of code if condition is False

```
if Boolean expression:
    block of code
else:
    block of code
```

More branching (out on a limb)

- Compound conditionals

```
if x<y and x<z:
    print('x is least‘)
elif y < z:
    print('y is least‘)
else:
    print('z is least‘)
```

- Nested conditionals

```
if x%2 == 0:
    if x%3 == 0:
        print('Divisible by 2 and 3‘)
    else:
        print('Divisible by 2 and not by 3‘)
elif x%3 == 0:
    print('Divisible by 3 and not by 2‘)
```

Strings and input

- Strings are an example of a **non-scalar** type
- Examples: ‘1bc’, ‘123’, ‘I501 is awesome!’
- Operators + and * are **overloaded**
  - “adding” two strings results in a **concatenated** string
  - “multiplying” two strings results in a **repeating** string
- Other operations mixing numbers and strings are not allowed – this is due to **type checking**

Operations on strings

- Length of a string **len()** *function*
- **Indexing** is used to get a substring
  - 0 based – first character in the string is position 0
  - Negative numbers count from the end of the string
- **Slicing** extracts a substring using start and end indices
  - The end index is 1 past the end of the slice
  - If the start index is missing the default is 0
  - If the end index is missing the default is the end of the string
  - ‘abc’[0:3] == ‘abc’

Input

- The **input** function prompts for a value and returns a string
  - `name = input(First Name?: ‘)`
- Use **type casting** to return a different type
  - `number = int(input(‘Pick a number between 1 and 10: ‘))`

Iteration

- Repeat a block of code a certain number of time or until a condition is reached
  - Looping or iteration
  - Loop terminates when a conditional is false
    - Often involves a “decrementing function”
  - OR when the loop is *broken*

```
# Square an integer, the hard way
x = 3
ans = 0
itersLeft = x
while (itersLeft != 0):
    ans = ans + x
    itersLeft = itersLeft - 1
print(str(x) + '*' + str(x) + ' = ' + str(ans))
```

For Loops

- Iterates across a sequence of objects
- The sequence can be any type of object
- Loop executed for first object in the sequence
  - Until last object is processed
  - Or a **break** statement is executed
- Most common form is a Python **range**
- The range is evaluated at the beginning of the loop

Range

- Starting value
- Ending value (exceeds the last element)
- Step

- **range**(*start*, *end*, *step*)
- **range**(*start*, *end*)
- **range**(*end*)

Nested **for** loop example

```
x = 4
for j in range(x)
    for i in range(x)
        print(j, i)
        x = 2
```

```
0 0
0 1
0 2
0 3
1 0
1 1
2 0
2 1
3 0
3 1
```