Python Lab: Control Statements

Conditionals, Loops, Iterations

Proteomics Informatics, Spring 2014

Week 4

18th Feb, 2014

Himanshu.Grover@nyumc.org

Recap

- Data types (int, float, str etc.) and values (2, 4.6, “ATCG”)
- Variables (x = 10, y = “ATCG”)
- Data structures (lists, dictionaries)
- Operations
- Simple python statements and expressions

Today

- Control statements:
  - Control the flow of the program/code
  - Conditionals and Loops

- Allow building complex logic in programs

Eclipse/Pydev setup ??

‘if/else’ Conditionals

- **Idea:**
  - Test for condition & take appropriate action
  - conditional execution vs. branching

- **Basic structure**

(Conditional Execution)

If <*expression*>:
    <***statements’ block***>

(Alternative execution)

if <expression$_1$>:
    <statements$_1$ block>
else:
    <statements$_2$ block>

- **Ex.**

```
x = 3
if x % 4 == 0:      ## (modulus operator)
    print “x is divisible by 4”

if x < 5:
    print “low expression”
else:
    print “high expression”
```

Expressions must evaluate to True or False

- **Relational operators:**
  - **x == y** (x equal y?)
  - **x > y** (x greater than y?)
  - **x < y** (x less than y?)
  - **x >= y** (x greater than or equal to y?)
  - **x <= y** (x less than or equal to y?)
  - **x != y** (x not equal to y?)
- **Membership (**x in y)
  - "PEP" in "PEPTIDE", "PET" not in "PEPTIDE"
  - 2 in [1,2,3,4,5]
- **Type-specific, built-in vs. user-defined:**
  - "PEPTIDE".isupper()
- **Logical or boolean operators (**and, or, not)
  - help build more complex logic
  - x == y **and** x != z
  - x > 5 **or** x < -5
  - x > 5 **and** x < 10
  - **not (**x > 5 **and** x < 10**)**

Other 'if/else' Structures

- **Multiple tests (chaining)**

if <expression$_1$>:
    <statements$_1$ block>
elif <expression$_2$>:
    <statements$_2$ block>
else:
    <statements$_3$ block>

Ex.
```
x = 3
if x < 3:
    print "low"
elif x < 7:
    print "medium"
else:
    print "high"
```

- **Nested conditionals**

if <expression$_1$>:
    if <expression$_2$>:
        <statements$_2$ block>
    else:
        <statements$_3$ block>

Ex.
```
disease = True
x = 3
if disease == True:
    if x > 0 and x <= 4:
        print "give drug A"
    elif x > 7 and x <= 10:
        print "give drug B"
    else:
        print "no drug"
```

‘for’ Loops

- **Idea:** repeat action(s) for each item in a sequence or any other iterable object (**Ex.** str, list, dict, set etc.)

- **Basic structure**

  for <*item*> in <*sequence*>:

  <statements’ block>

‘for’ loop: Access Pattern 1

- Sequence scans

- **Ex.** numList = [1,2,3,4]

```
sum = 0
prod = 1
for num in numList:
    sum = sum + num
    prod = prod*num
print “sum: %d\tprod: %d”%(sum, prod)
```

‘for’ loop: Access Pattern 2

- Counter/index: **range** function
  - Greater flexibility for more specialized operations

- **Ex.**

```
numList = [1,2,3,4]
sum = 0
prod = 1
for pos in range(len(numList)):
    sum = sum + numList[pos]
    prod = product*numList[pos]
print “sum: %d\tprod: %d”%(sum, prod)
```

HW Assignment

- Given a peptide sequence and charge state, compute its m/z

Next Class

- More examples of loops
- Write simple functions