

# CS 133 - Introduction to Computational and Data Science

Instructor: Renzhi Cao  
Computer Science Department  
Pacific Lutheran University  
Spring 2017



# Introduction to Python II

- In the previous class, you have learned lists, dictionary, tuples, and how to write functions.

sum(1,2)

Pass the value to the parameters x and y

```
def sum(x,y) :  
    print x  
    print y  
    result = x + y  
    return result
```

sum(3,4)  
sum(4,5)

# The function practice

- Check the website and implement functions

# Booleans

- 0 and None are false
- Everything else is true
- True and False are aliases for 1 and 0 respectively

# Control flow

## Things that are False

- The boolean value False
- The numbers 0 (integer), 0.0 (float) and 0j (complex).
- The empty string "".
- The empty list [], empty dictionary {} and empty set set().

## Things that are True

- The boolean value True
- All non-zero numbers.
- Any string containing at least one character.
- A non-empty data structure.

# Control flow

There are cases that you want specific block of code to be functional when some condition is true.

- User type 'yes', do calculation, type 'no', quit program
- When temperature is higher than 100 degree, print 'hot'.
- When your bank account has 0 balance, user cannot withdraw any money.

# If statement

The code we have seen before is “always” executed. How would we create cases in which only some code is executed?

- **if expression:** # expression is boolean type  
do something when expression is True  
[else:] # this is optional



# If statement

```
>>> smiles = "BrC1=CC=C(C=C1)NN.Cl"
>>> bool(smiles)
True
>>> not bool(smiles)
False
>>> if not smiles:
...     print "The SMILES string is empty"
...
The "else" case is always optional
```

# If statement

```
>if x% 2 == 0:  
    print 'x is even'  
else:  
    print 'x is odd'
```

**What is the % doing here?**

# If statement

```
>if x == y:  
    print 'x and y are equal'  
else:  
    if x < y:  
        print 'x is less than y'  
    else:  
        print 'x is greater than y'
```

**Observe the use of indentation**

# "elif"

```
>>> mode = "absolute"
>>> if mode == "canonical":
...     smiles = "canonical"
...     elif mode == "isomeric":
...         smiles = "isomeric"
...     elif mode == "absolute":
...         smiles = "absolute"
...     else:
...         raise TypeError("unknown mode")
...
>>> smiles
'absolute'
```

>>>  
“raise” is the Python way to raise exceptions

# Practices

1. Get user's score, save it as variable `score`.
2. print 'A' for score in  $[90,100]$ , 'B' for  $[80,90)$ , 'C' for  $[70,80)$ , 'D' for rest of scores.

# Boolean logic

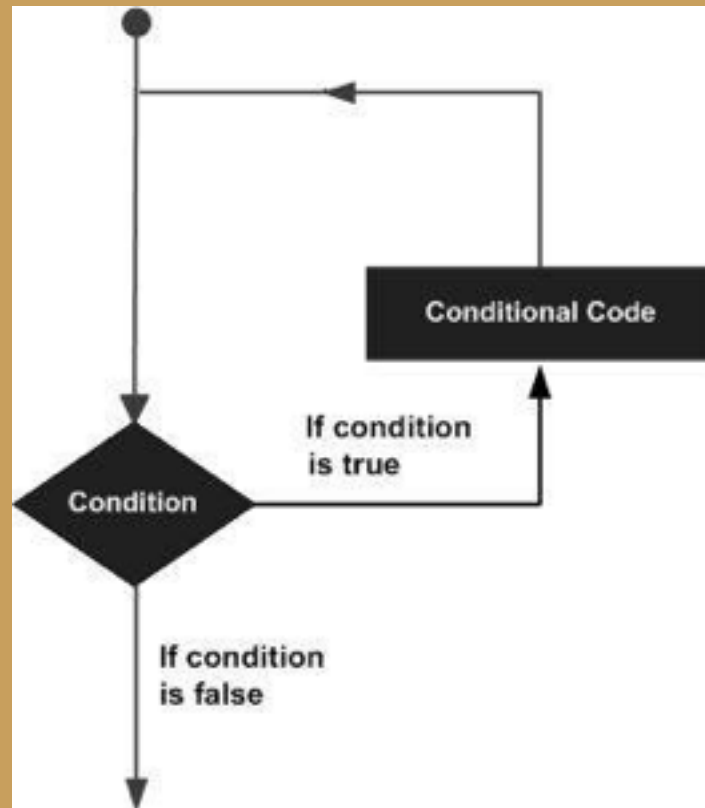
Python expressions can have “and”, “or”:

```
if(a <= 10 and b >= 10 or a == 100 and b!= 5):  
    print “Hello”
```

```
if( 3 <= a <= 100):  
    print “great!”
```

# For statement

Python use “for” as keyword to handle loops.



# For statement

- `>>> names = ["cao", "python"]`
- `>>> for name in names:`  
    `print name`



# For statement

```
data = [ ("C20H20O3", 308.371),  
        ("C22H20O2", 316.393),  
        ("C24H40N4O2", 416.6),  
        ("C14H25N5O3", 311.38),  
        ("C15H20O2", 232.3181)]
```

```
for (formula, mw) in data:  
    print "The molecular weight of %s is %s" % (formula, mw)
```

The molecular weight of C20H20O3 is 308.371

The molecular weight of C22H20O2 is 316.393

The molecular weight of C24H40N4O2 is 416.6

The molecular weight of C14H25N5O3 is 311.38

The molecular weight of C15H20O2 is 232.3181

# Loop Control Statements

<b>break</b>	Jumps out of the closest enclosing loop
<b>continue</b>	Jumps to the top of the closest enclosing loop
<b>pass</b>	Does nothing, empty statement placeholder

# Break and continue in loop

```
>>> for value in [3, 1, 4, 1, 5, 9, 2]:
...     print "Checking", value
...     if value > 8:
...         print "Exiting for loop"
...         break
...     elif value < 3:
...         print "Ignoring"
...         continue
...     print "The square is", value**2
... 
```

# Range

- “range” creates a list of numbers in a specified range
- range([start,] stop[, step]) -> list of integers
- When step is given, it specifies the increment (or decrement).

```
>>> range(5)
```

```
[0, 1, 2, 3, 4]
```

```
>>> range(5, 10)
```

```
[5, 6, 7, 8, 9]
```

```
>>> range(0, 10, 2)
```

```
[0, 2, 4, 6, 8]
```

How to get every second element in a list?

```
for i in range(0, len(data), 2):
```

```
    print data[i]
```

# while

Similar to for, the usage is:

while expression:

    always do when expression is True.

# while

>while True:

```
    line = raw_input(' > ')
```

```
    if line == 'done':
```

```
        break
```

```
    print line
```

```
print 'Done!'
```

**How to expand to accept other words?**

# while

Set a condition for the loop to end

```
>def sequence( n):
```

```
    while n != 1:
```

```
        print n,
```

```
        if n% 2 == 0: # n is even
```

```
            n = n/ 2
```

```
        else: # n is odd
```

```
            n = n* 3 + 1
```

# Practice

Fermat's Last Theorem says that there are no integers  $a$ ,  $b$ , and  $c$  such that

$$a^n + b^n = c^n$$

for any values of  $n$  greater than 2.

1. Write a function named `check_fermat` that takes four parameters —  $a$ ,  $b$ ,  $c$  and  $n$  — and that checks to see if Fermat's theorem holds. If  $n$  is greater than 2 and it turns out to be true that

$$a^n + b^n = c^n$$

the program should print, "Holy smokes, Fermat was wrong!" Otherwise the program should print, "No, that doesn't work."



# Practice

The built-in function `eval` takes a string and evaluates it using the Python interpreter. For example:

```
>>> eval('1 + 2 * 3')
7
>>> import math
>>> eval('math.sqrt(5)')
2.2360679774997898
>>> eval('type(math.pi)')
<type 'float'>
```

Write a function called `eval_loop` that iteratively prompts the user, takes the resulting input and evaluates it using `eval`, and prints the result.

It should continue until the user enters `'done'`, and then return the value of the last expression it evaluated.