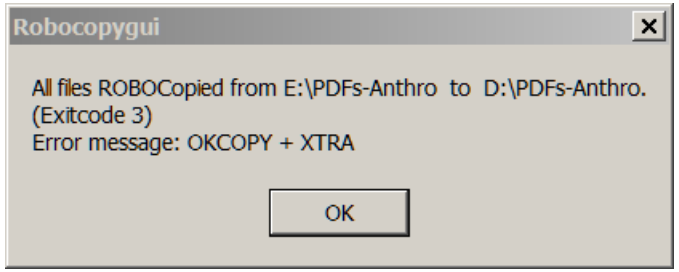
DATA 520
Lecture 12
Review
Program Integration and Planning

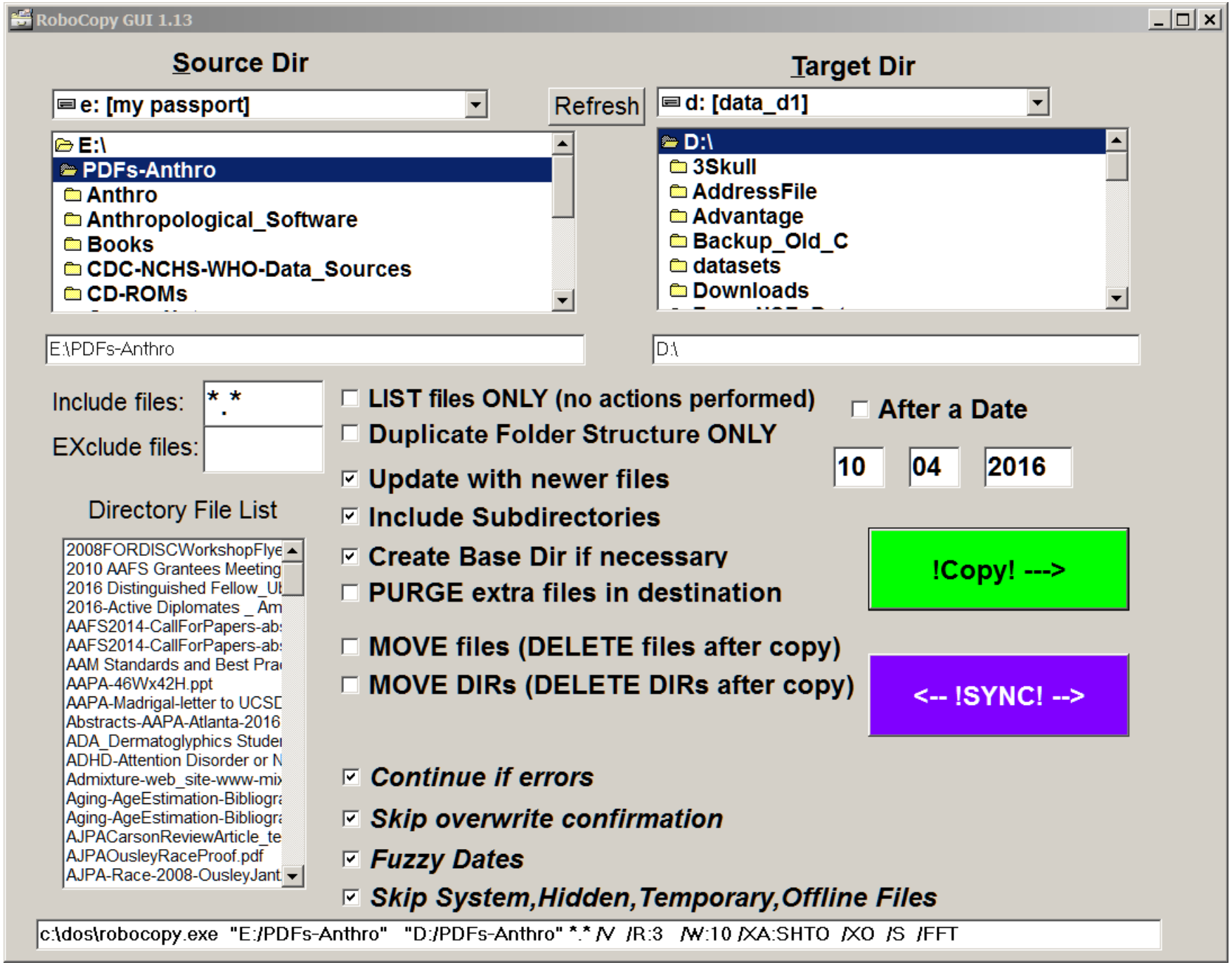
What should the program do?

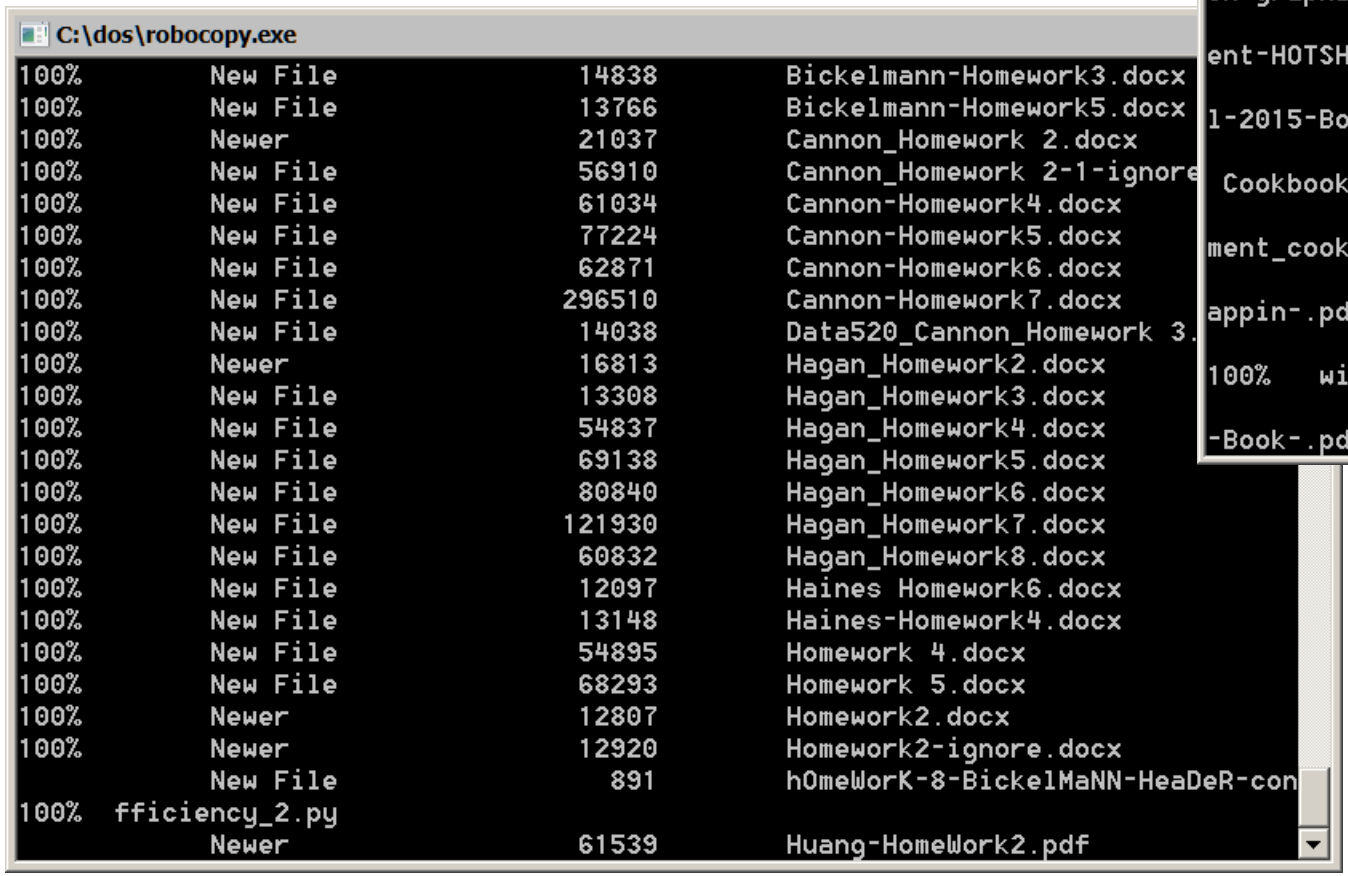
How to do it?

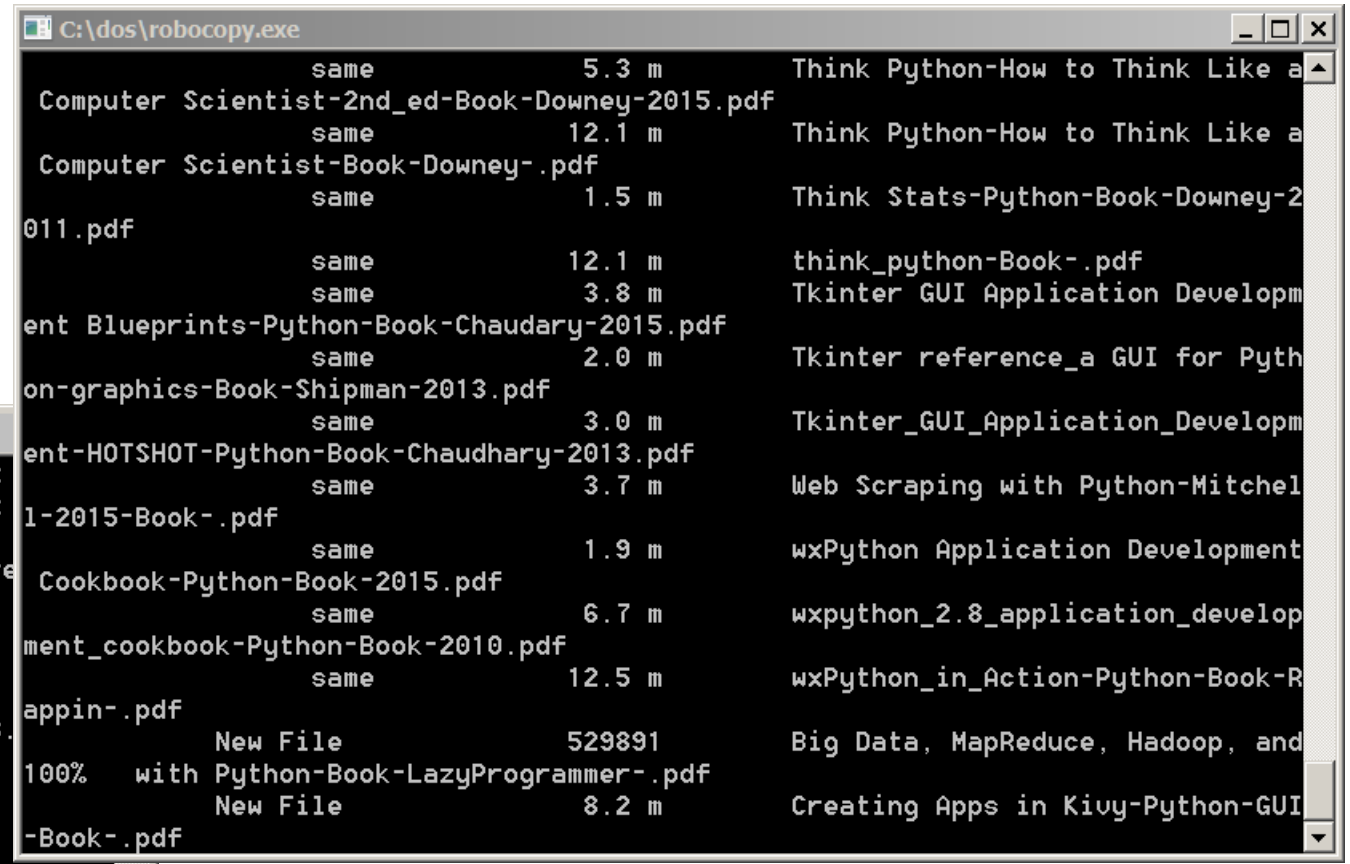
Application Monday

Robocopy, GUI

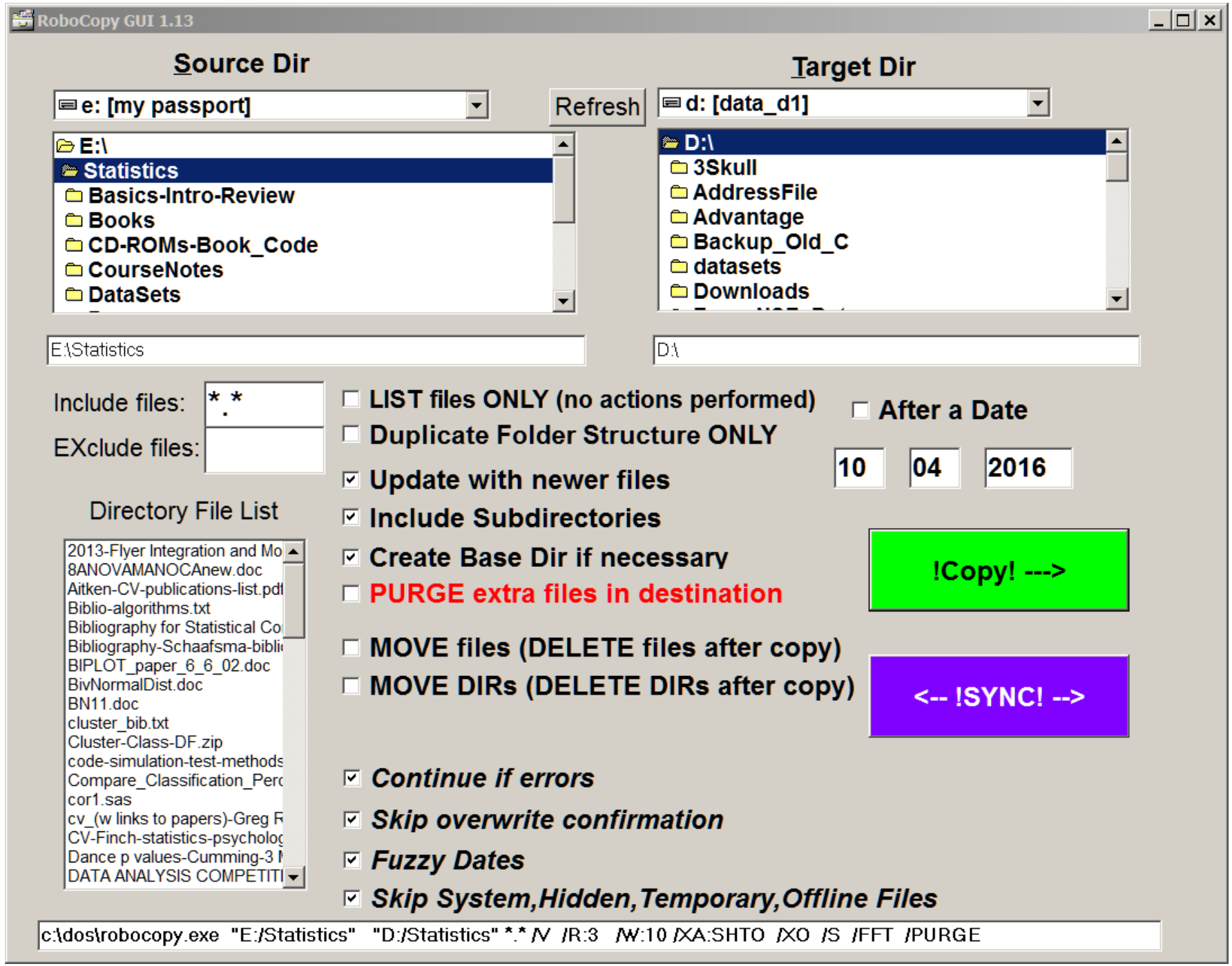


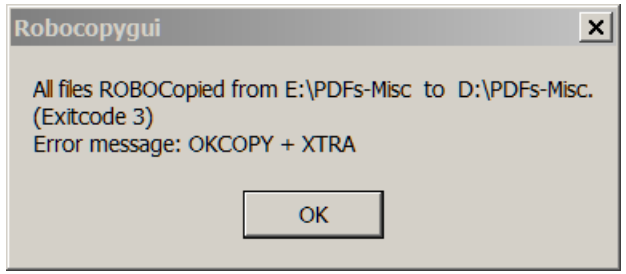







```
robocopy  %1\2017-Fall\ E:\Mercyhurst\Courses\2017-Fall   /E  /XO /FFT /R:3   /W:5  /XA:SHTO
```


RoboCopy GUI 1.13
Source Dir
e: [my passport]
Refresh
Target Dir
d: [data_d1]
E:\
Statistics
Basics-Intro-Review
Books
CD-ROMs-Book_Code
CourseNotes
DataSets
D:\
3Skull
AddressFile
Advantage
Backup_Old_C
datasets
Downloads
E:\Statistics
D:\
Include files:
.
EXclude files:
Directory File List
2013-Flyer Integration and Mo
8ANOVAMANOCAnew.doc
Aitken-CV-publications-list.pdf
Biblio-algorithms.txt
Bibliography for Statistical Co
Bibliography-Schaafsma-biblio
BIPLOT_paper_6_6_02.doc
BivNormalDist.doc
BN11.doc
cluster_bib.txt
Cluster-Class-DF.zip
code-simulation-test-methods
Compare_Classification_Perc
cor1.sas
cv_(w links to papers)-Greg R
CV-Finch-statistics-psycholog
Dance p values-Cumming-3 M
DATA ANALYSIS COMPETITI
LIST files ONLY (no actions performed)
Duplicate Folder Structure ONLY
Update with newer files
Include Subdirectories
Create Base Dir if necessary
PURGE extra files in destination
After a Date
10
04
2016
!Copy! --->
MOVE files (DELETE files after copy)
MOVE DIRs (DELETE DIRs after copy)
<-- !SYNC! -->
Continue if errors
Skip overwrite confirmation
Fuzzy Dates
Skip System,Hidden,Temporary,Offline Files
c:\dos\robocopy.exe "E:/Statistics" "D:/Statistics" *.* /V /R:3 /W:10 /XA:SHTO /XO /S /FFT /PURGE



RoboCopy GUI 1.13

Source Dir

e: [my passport]

Refresh

Target Dir

d: [data_d1]

E:\
PDFs-Misc
Collections
Computers
Magazines
Recipes from Marius
REFDocs

D:\
3Skull
AddressFile
Advantage
Backup_Old_C
datasets
Downloads

E:\PDFs-Misc

D:\

Include files: *.*

EXclude files:

Directory File List

10 Modern Statistical Concep
12deadly.pdf
12DeadlyManagerSins.pdf
13 Things that dont make sen
14WorstRetirementMistakes.
15 funniest travel books ever
20_Thinking_Tools_Collabora
25 Essential Techniques Gas
3D_Computer_Graphics_A_M
50 Articles about Hadoop and
50 Timeless Scientists-Book-
50GermanInventions-stars.pd
A Book Apart Publishing HTMl
A Field Guide for Science Wri
A Hole in the Head-Neuroscie
A Physicist Experiments With
A Pocket Popper-Book-Miller-
A Serbian Sokal Authors spo

- [ ] LIST files ONLY (no actions performed)
- [ ] Duplicate Folder Structure ONLY
- [x] Update with newer files
- [x] Include Subdirectories
- [x] Create Base Dir if necessary
- [x] PURGE extra files in destination
- [ ] MOVE files (DELETE files after copy)
- [ ] MOVE DIRs (DELETE DIRs after copy)

- [ ] After a Date

10 04 2016

!Copy! --->

<-- !SYNC! -->

- [x] *Continue if errors*
- [x] *Skip overwrite confirmation*
- [x] *Fuzzy Dates*
- [x] *Skip System,Hidden,Temporary,Offline Files*

c:\dos\robocopy.exe "E:/PDFs-Misc" "D:/PDFs-Misc" *.* /V /R:3 /W:10 /XA:SHTO /XO /S /FFT /PURGE

Python is a special calculator

```
# integer division, with a DOWNWARD truncated integer answer (floor)
# <= the integer part of answer
53 / 24
2.2083333333333335
53 // 24
2
4 // 2
2
# bizarre but consistent: negative numbers
53 // -24
-3
-53 // 24
-3
```

Python is a special calculator

```
# modulo and floor using floats return a float
53.222 // 24.994757
2.0

53.222 % 24.994757
3.2324860000000015

# Exponents use **
# 6 to the third power: 6³ (R, Excel: 6^3)
6 ** 3
216
6 ** 0.5  # what is this asking for?
2.449489742783178
```

Python is a special calculator

```
# be careful with negation (minus sign)
-2 ** 4
-16
```

which means Python does the exponentiation BEFORE the negation

```
-(2 ** 4)  # how Python sees the above
-16
#what we wanted was this:
(-2) ** 4

# maybe it will not come up often:
val = -2
val ** 4
16
```

Python and computer precision

```
# but errors in precision do not grow
700/3
233.33333333333334
7000/3
2333.3333333333335

7000000000000/3
2333333333333.3335

70000000000000000000000000000000000000000000000000000000000000000 / 3
2.3333333333333333e+60
```

Python and computer precision

```
# but precision can be tricky when testing for equivalence
2 / 3 + 1  # = 5/3
1.6666666666666665
5 / 3
1.6666666666666667
5 / 3  - (2/3 + 1)  # subtract
2.220446049250313e-16

10000000000 + 0.00000000001
10000000000.0
```

It looks like nothing happened!

Gries: Add from smallest to largest to minimize error

(computers: single, double, and extended precision)

Variables in Python: Automatic type

```
# To go from Celsius to Fahrenheit:
# F = 9/5 * C + 32
# C = (F - 32) * 5/9

degrees_celsius = 26  # degrees_celsius or CD or CelDeg
9 / 5 * degrees_celsius + 32
78.80000000000001
DF = 9 / 5 * degrees_celsius + 32  # assign variable DF to the result
78.80000000000001
degrees_celsius = 0  # change for another calculation
9 / 5 * degrees_celsius + 32
32.0
DF    # has not changed value
78.80000000000001
```

Useful Numeric Functions

```
int(34.65) # return integer, truncated
34
int(-4.3)
-4
float(21) # convert a number (integer or float) into a float
21.0
float(3e+15)  # scientific notation returns number
3000000000000000.0
int(3e+15)
3000000000000000
round (2.5435345) # rounds a number, default is integer
3
round (2.5435345,1) # rounds to one decimal place; 0 returns a float
2.5
```

Functions are the building blocks of most programs

Normal format for calling a function:

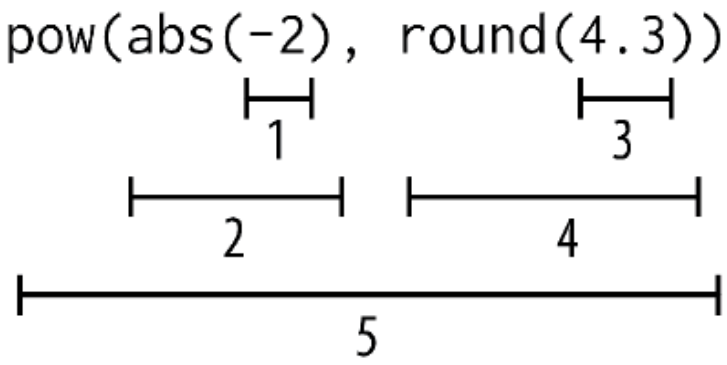
<function name>**(**<argument(s)>**)**

functions return something (or modify something)

```
abs(-9) # absolute value
9
```

nested functions processed from left to right

```
pow(abs(-2), round(4.3))# = 2**4
16
```



Defining functions: local variables

```
# Calculate quadratic formula, ax^2+ bx + c given a,b,c,x
# return can involve a calculation
def quadratic(a, b, c, x):
    first = a * x ** 2
    second = b * x
    third = c
    return first + second + third
```

a,b,c, and x are **parameters**: only valid within the function

(**arguments** are provided to the function)

first, second, and third are local variables: only valid within the function

Defining functions more formally

```
def convert_to_celsius(fahrenheit):
    """ (number) -> float

    Return the number of Celsius degrees equivalent to fahrenheit degrees.


    >>> convert_to_celsius(75)
    23.8888888888889
    """
    return (fahrenheit - 32.0) * 5.0 / 9.0
```

function header

notice three double quotes here start the **docstring** :
starts with parameter and return formats (type contract)

number = integer or float

description

examples

body (the code that does something!)

fahrenheit in this example is a parameter

Defining functions more formally

Days difference macro

```
def days_difference(day1, day2):
    """ (int, int) -> int

    Return the number of days between day1 and day2, which are
    both in the range 1-365 (thus indicating the day of the
    year).

    >>> days_difference(200, 224)
    24
    >>> days_difference(50, 50)
    0
    >>> days_difference(100, 99)
    -1
    """
    return day2 - day1
```

Functions

Built-in Functions				
abs()	divmod()	input()	open()	staticmethod()
all()	enumerate()	int()	ord()	str()
any()	eval()	isinstance()	pow()	sum()
basestring()	execfile()	issubclass()	print()	super()
bin()	file()	iter()	property()	tuple()
bool()	filter()	len()	range()	type()
bytearray()	float()	list()	raw_input()	unichr()
callable()	format()	locals()	reduce()	unicode()
chr()	frozenset()	long()	reload()	vars()
classmethod()	getattr()	map()	repr()	xrange()
cmp()	globals()	max()	reversed()	zip()
compile()	hasattr()	memoryview()	round()	__import__()
complex()	hash()	min()	set()	
delattr()	help()	next()	setattr()	
dict()	hex()	object()	slice()	
dir()	id()	oct()	sorted()	

Strings and floats

```
# what about floats?;
xr1 = 23.584564;
print('The answer is ' + format(xr1,'.2f'));
The answer is 23.58
# use str(), int(), float(), format() to convert types

>>> int('a')
Traceback (most recent call last):
File "<stdin>", line 1, in <module>
ValueError: invalid literal for int() with base 10: 'a'
>>> float('b')
Traceback (most recent call last):
File "<stdin>", line 1, in <module>
ValueError: could not convert string to float: 'b'
```

Strings

```
# escape character (\) and sequence
len('\')
1
len('\'')  # or len("\"") or len("\"")
1
len ('it\'s') # it's
4
Other escape sequences
\' Single quote
\" Double quote
\\ Backslash - useful for directories c:\\program files\\python
\t Tab
\n Newline
\r Carriage return
```

Getting Strings as Input

```
species = input('Enter a species name:')
# we can now use that variable, but we MUST use +
pop = input('Enter the number of organisms represented by ' +  species + ': ')
pop = pop + 1
print ('The population of ', species, ' is ' , pop)
```

Boolean Operators

Combinations of and + or: Always use parentheses with OR!

```
b1 = False
b2 = False
(b1 and not b2) or (b2 and not b1)
```

```
False
```

```
b1 and not b2 or b2 and not b1
```

```
False
```

```
not b1 and not b2
```

```
True
```

```
not b1 or not b2 or (1/0)  # 1/0 is undefined
```

```
True
```

NOTE: statements are not completely evaluated - as soon as one answer is clear, it moves on

Boolean logic

More transparent comparisons (note the == and !=):

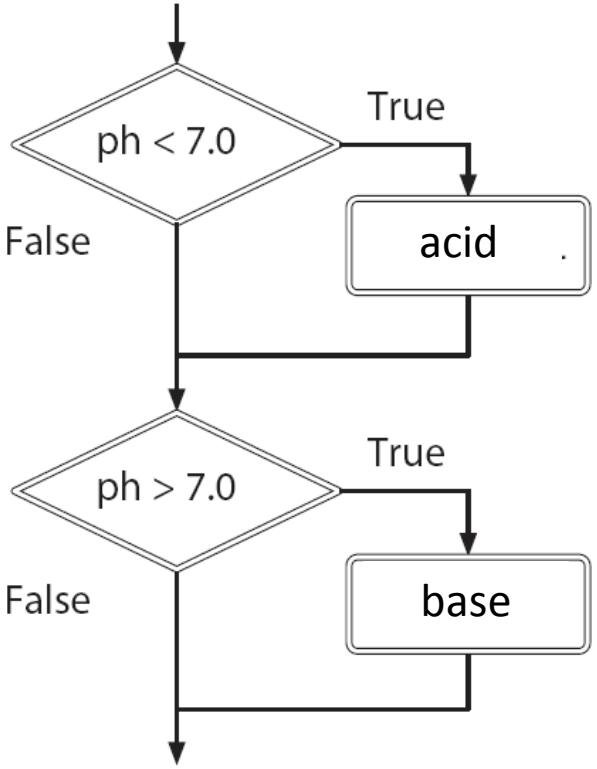
Symbol	Operation
>	Greater than
<	Less than
>=	Greater than or equal to
<=	Less than or equal to
==	Equal to
!=	Not equal to

Boolean logic: IF

```
if ph < 7.0:
    print(ph, "is acidic.")                  # 4 spaces!
    print('Be careful with that acid!!!') # 4 spaces!
```

```
# new addition to code
if ph < 7.0:
    print(ph, " is acidic.")
if ph > 7.0:
    print(ph, " is basic.")
```

problem?

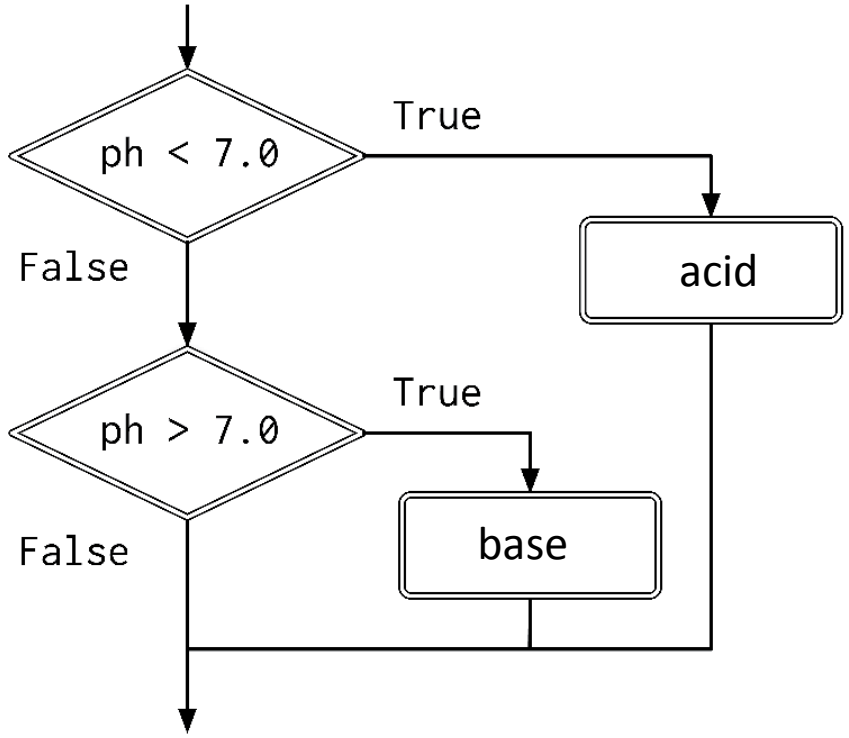


Boolean logic: IF - ELIF (else if) branching

```
# BETTER addition to code
# if.. (then do this)  else if (then do this)
if ph < 7.0:
    print(ph, " is acidic.")
elif ph > 7.0:
    print(ph, " is basic.")

In shell it looks funky:
>>> if ph < 7.0:
    print(ph, " is acidic.")
elif ph > 7.0:
    print(ph, " is basic.")

5.0  is acidic.
```



Boolean logic: Branching (elif and else:)

```
# use else to cover ALL cases
compound = input('Enter the compound: ')
Enter the compound: H2SO4
if compound == "H2O":
    print("Water")
elif compound == "NH3":
    print("Ammonia")
elif compound == "CH4":
    print("Methane")
else:  print("Unknown compound")

^^^^ notice formatting!!!!
```

```
>>> compound = input('Enter the compound: ')
Enter the compound: H2SO4
>>> if compound == "H2O":
...     print("Water")
... elif compound == "NH3":
...     print("Ammonia")
... elif compound == "CH4":
...     print("Methane")
... else:
...     print("Unknown compound")
...
Unknown compound
```

```
>>> if compound == "H2O":
    print("Water")
    elif compound == "NH3":
    print("Ammonia")
    elif compound == "CH4":
    print("Methane")

SyntaxError: invalid syntax
>>> if compound == "H2O":
    print("Water")
elif compound == "NH3":
    print("Ammonia")
elif compound == "CH4":
    print("Methane")
else:  print("Unknown compound")

Unknown compound
```

Boolean logic: Logic power

Risk for heart disease based on BMI and Age

notice these MUST be T or F

```
young = age < 45
slim = bmi < 22.0
if young and slim:
    risk = 'low'
elif young and not slim:
    risk = 'medium'
elif not young and slim:
    risk = 'medium'
elif not young and not slim:
    risk = 'high'
```

		Age	
		<45	≥45
BMI	<22.0	Low	Medium
	≥22.0	Medium	High

Modules

A Module is a collection of functions (often algorithms)
and variables (usually constants) in a single file

Modules are usually based on a theme

- String handler, or searcher, or parser
- Statistical functions (general), or Matrix operations, or Machine Learning
- Time and Date functions
- Encryption / Decryption / Compression (ZIP, tar, etc.)
- Python things (version, etc.)
- SYSTEM things (time, OS, OS version, disk space, etc.)

Modules

To use a module, import it.

```
import math  # math things based on a C library
help(math)
```

- take a look

e **(2.718281828459045),** ***pi*** **(π) (3.141592653589793)**

sin, cos, convert radians to degrees, different rounding functions, hypotenuse,

```
math.ceil(x)   This is the smallest integer >= x.
math.floor(x)  This is the largest integer <= x.
inf, nan
```

Modules

Python's built-in functions

```
dir(__builtins__)
```

```
['ArithmeticError', 'AssertionError', 'AttributeError', 'BaseException', 'BlockingIOError', 'BrokenPipeError',
'BufferError', 'BytesWarning', 'ChildProcessError', 'ConnectionAbortedError', 'ConnectionError',
'ConnectionRefusedError', 'ConnectionResetError', 'DeprecationWarning', 'EOFError', 'Ellipsis',
'EnvironmentError', 'Exception', 'False', 'FileExistsError', 'FileNotFoundError', 'FloatingPointError',
'FutureWarning', 'GeneratorExit', 'IOError', 'ImportError', 'ImportWarning', 'IndentationError', 'IndexError',
'InterruptedError', 'IsADirectoryError', 'KeyError', 'KeyboardInterrupt', 'LookupError', 'MemoryError',
'NameError', 'None', 'NotADirectoryError', 'NotImplemented', 'NotImplementedError', 'OSError',
'OverflowError', 'PendingDeprecationWarning', 'PermissionError', 'ProcessLookupError', 'RecursionError',
'ReferenceError', 'ResourceWarning', 'RuntimeError', 'RuntimeWarning', 'StopAsyncIteration', 'StopIteration',
'SyntaxError', 'SyntaxWarning', 'SystemError', 'SystemExit', 'TabError', 'TimeoutError', 'True', 'TypeError',
'UnboundLocalError', 'UnicodeDecodeError', 'UnicodeEncodeError', 'UnicodeError', 'UnicodeTranslateError',
'UnicodeWarning', 'UserWarning', 'ValueError', 'Warning', 'WindowsError', 'ZeroDivisionError', '_',
'__build_class__', '__debug__', '__doc__', '__import__', '__loader__', '__name__', '__package__', '__spec__',
'abs', 'all', 'any', 'ascii', 'bin', 'bool', 'bytearray', 'bytes', 'callable', 'chr', 'classmethod',
'compile', 'complex', 'copyright', 'credits', 'delattr', 'dict', 'dir', 'divmod', 'enumerate', 'eval', 'exec',
'exit', 'filter', 'float', 'format', 'frozenset', 'getattr', 'globals', 'hasattr', 'hash', 'help', 'hex',
'id', 'input', 'int', 'isinstance', 'issubclass', 'iter', 'len', 'license', 'list', 'locals', 'map', 'max',
'memoryview', 'min', 'next', 'object', 'oct', 'open', 'ord', 'pow', 'print', 'property', 'quit', 'range',
'repr', 'reversed', 'round', 'set', 'setattr', 'slice', 'sorted', 'staticmethod', 'str', 'sum', 'super',
'tuple', 'type', 'vars', 'zip']
```

Modules

Defining your own modules

save as temperature.py

then import

```
print ('temperature module loaded and ready!')

def convert_to_celsius(fahrenheit):
    """ (number) -> float

    Return the number of Celsius degrees equivalent to fahrenheit
degrees.

    >>> convert_to_celsius(75)
    23.88888888888889
    """
    return (fahrenheit - 32.0) * 5.0 / 9.0

def above_freezing(celsius):
    """ (number) -> bool

    Return True iff temperature celsius is above freezing.

    >>> above_freezing(5.2)
    True
    >>> above_freezing(-2)
    False
    """

    return celsius > 0
```

A module is loaded only once

Modules

Online: google search or

https://docs.python.org/3/library/index.html

```
download
# high precision math - extended digits (1,000!)
from mpmath import mp
mp.dps = 50
print(mp.quad(lambda x: mp.exp(-x**2), [-mp.inf, mp.inf]) ** 2)
3.1415926535897932384626433832795028841971693993751
```

With so many modules, how can we install them? with the help of a module!

```
# At command prompt: (Choose Start | Run | type in 'cmd' )
cd C:\Users\sousley\AppData\Local\Programs\Python\Python36-32
python -m pip install -U pip setuptools
# import matplotlib
python -m pip install matplotlib
# import mpmath
python -m pip install mpmath
```

Methods

A Method is like a function bound to a data item (an object of a certain type)

help(<<bool, float, int, or str>>)

Strings - many many functions

```
capitalize(...)
| S.capitalize() -> str
||
Return a capitalized version of S, i.e. make the first character
| have upper case and the rest lower case.
```

```
str.capitalize('mercyhurst')
'Mercyhurst'
# shorter form - data linked to method
'mercyhurst'.capitalize()
'Mercyhurst'
```

Methods

```
str.lower('SIStErs oF')
'sisters of'
str.upper('mercy')
'MERCY'
'Patricia'.swapcase()
'pATRICIA'
```

*** underlying string is unchanged ***

```
Boolean results:
str.islower('Erie')
False
str.isupper('PA,USA')
True
```

Methods

Flexible find a substring: return int

str.find(s)

str.find(s, beg)

str.find(s, beg, end)

help(str.find)

```
Help on method_descriptor:
find(...)
    S.find(sub[, start[, end]]) -> int

    Return the lowest index in S where substring sub is found,
    such that sub is contained within S[start:end].  Optional
    arguments start and end are interpreted as in slice notation.

    Return -1 on failure. (not found)
```

String Methods

Using str.find

```
'This is an example of a string.'.find('string')
24
This is an example of a string.
12345678901234567890123456789
What gives?
Python starts counting with zero.
This is an example of a string.
01234567890123456789012345678
```

```
'This is an example of a string.'.find('stringl')
-1
```

which means it was not found **(NOT 0!)**

```
'This is an example of a string.'.find('This')
0
```

doctest

Using doctest (built-in)

```
def median(pool):
    '''Statistical median to demonstrate doctest.
    >>> median([2, 9, 9, 7, 9, 2, 4, 5, 8])
    7
    >>> median([2, 9, 9, 9, 2, 4, 5, 8])
    6.5
    '''
    copy = sorted(pool)
    size = len(copy)
    if size % 2 == 1:
        return copy[(size - 1) // 2]
    else:
        return (copy[size//2 + 1] + copy[size//2]) / 2
if __name__ == '__main__':
    import doctest
    doctest.testmod()
```

```
File "C:/Users/sousley/AppData/Local/Programs/Python/Python36-32/median.py", line 5, in
Failed example:
    median([2, 9, 9, 9, 2, 4, 5, 8])
Expected:
    6.5
Got:
    8.5
**********************************************************************
1 items had failures:
   1 of   2 in __main__.median
***Test Failed*** 1 failures.
```

Lists

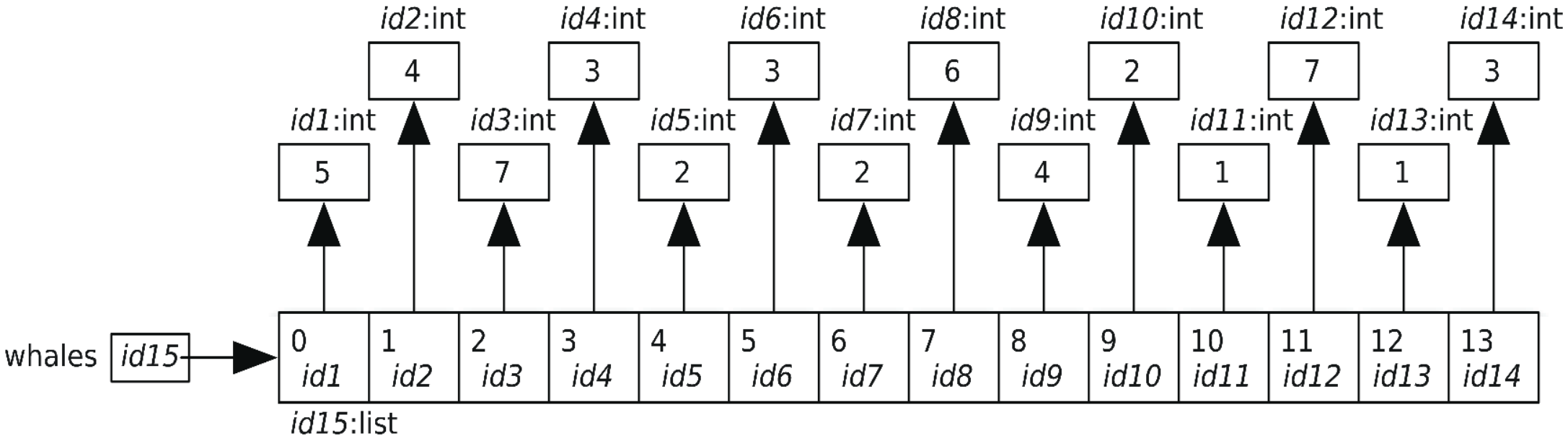
Lists store data sequentially

Lists use brackets

whales contains daily counts of whale sightings

```
whales = [5, 4, 7, 3, 2, 3, 2, 6, 4, 2, 1, 7, 1, 3]
```

each piece of data is stored in memory



Lists

Lists start counting with zero (like most Python things)

```
whales[0:4] # the first four in the list, starts with 0 UP TO 4 (not included)
[5, 4, 7, 3]
# notice that a list type is returned
third_day_cnt = whales[2]

# one way to print
print('There were ', third_day_cnt , ' whales sighted on the third day.')
There were  7  whales sighted on the third day.

# a bit more control this way:
print('There were ' + str(third_day_cnt) + ' whales sighted on the third day.')
There were 7 whales sighted on the third day.
```

Loops

Loops repeat things any number of times

Life without Loops:

```
# 4 different speeds to be shown and converted
velocities = [0.0, 9.81, 19.62, 29.43]
print('Metric:', velocities[0], 'm/sec;  Imperial:', velocities[0] * 3.28, 'ft/sec')
print('Metric:', velocities[1], 'm/sec;  Imperial:', velocities[1] * 3.28, 'ft/sec')
print('Metric:', velocities[2], 'm/sec;  Imperial:', velocities[2] * 3.28, 'ft/sec')
print('Metric:', velocities[3], 'm/sec;  Imperial:', velocities[3] * 3.28, 'ft/sec')
```

Life with Loops:

```
for velocity in velocities:   # note: velocity not defined
    print('Metric:', velocity, 'm/sec; Imperial:', velocity * 3.28, 'ft/sec')
```

for *«each item»* **in** *«list»*:
    «code block»

Loops

for loops start with the first item and execute code through to the last item

```
step 1 out of 5 steps
step 2 out of 5 steps
step 3 out of 5 steps
step 4 out of 5 steps
step 5 out of 5 steps
```

Each time the code block is executed it is an iteration

```
# now try this code;
velocities = [0.0, 9.81, 19.62, 29.43]
speed = 2
for speed in velocities:   # note: speed IS defined, but overwritten
    print('Metric:', speed, 'm/sec')

print('Final(speed):', speed)
```

```
Metric: 0.0 m/sec
Metric: 9.81 m/sec
Metric: 19.62 m/sec
Metric: 29.43 m/sec
Final(speed): 29.43
```

speed at end has changed to last one in list

Loops

```
# coerce range to a list
list(range(10))
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
list(range(2))
[0, 1]

# range with 2 numbers: start value, stop value (NOT included)
list(range(1,5))
[1, 2, 3, 4]

# range with 3 numbers: start value, stop value (NOT included), step value
list(range(1,10,2))
[1, 3, 5, 7, 9]
#election years
list(range(1900,2020,4))
[1900, 1904, 1908, 1912, 1916, 1920, 1924, 1928, 1932, 1936, 1940, 1944, 1948, 1952, 1956, 1960,
   1964, 1968, 1972, 1976, 1980, 1984, 1988, 1992, 1996, 2000, 2004, 2008, 2012, 2016]
```

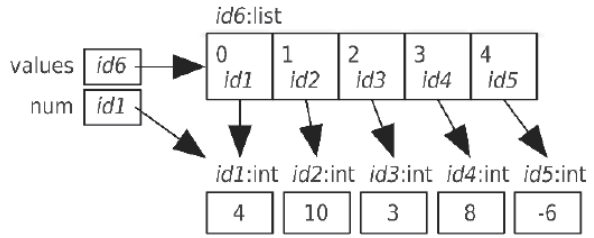
Loops

Lists are unaffected by loops

```
lvalues = [4, 10, 3, 8, -6]
for num in lvalues:
    num = num * 2
print(lvalues)
[4, 10, 3, 8, -6]
num
-12
```

```
lvalues = [4, 10, 3, 8, -6]
for num in lvalues:
    num = num * 2
    print(num)
8
20
6
16
-12
```

```
for num in values:
```



What if we want to change the values in a list?

We can't loop through numbers we are changing

We CAN loop through the indices of the list [0,1,2,3,4,5..]

```
list(range(len(lvalues)))  # len(lvalues) = 5; list(range(5))
[0, 1, 2, 3, 4]
```

Loops

We can loop through the indices of the list lvalues[0,1,2,3,4,5..]

```
list(range(len(lvalues)))  # len(lvalues) = 5; list(range(5)) = [0,1,2,3,4]
[0, 1, 2, 3, 4]
# so use code:
values = [4, 10, 3, 8, -6]
for i in range(len(values)):  # index number 0 to 4
    print(i, values[i])

values = [4, 10, 3, 8, -6]
for i in range(len(values)):   # index number 0 to 4
    values[i] = values[i] * 2  # index number 0 to 4
print(values)
```

Loops

Parallel processing using indices

```
# corresponding data by index: code
metals = ['Li', 'Na', 'K']
awts = [6.941, 22.98976928, 39.0983]
for i in range(len(metals)):
    print(metals[i], awts[i])
    #print(metals[i] + '\t' + str(awts[i]))
```

Note: using metals index means metal items <= awts; if not, IndexError: list index out of range

You might call for item 4 in one list but there are only 3.

Loops

Looping until a condition is reached or something happens: while

while *«expression»*:

«block»

In editor:

```
#rabbits reproduce!
rabbitcount = 2
while rabbitcount > 0:
    print(rabbitcount)
    rabbitcount = rabbitcount * 2
```

```
2
4
8
16
32
64
128...  (press Ctrl-C to stop)
```

Loops

Looping until a condition is reached or something happens: while

- based on user input (note the indents!)

```
text = ""
while text != "quit":
    text = input("Please enter a chemical formula H2O,NH3,CH4 (or 'quit' to exit): ")
    if text == "quit":
        print("exiting program...")
    elif text == "H2O":
        print("Water")
    elif text == "NH3":
        print("Ammonia")
    elif text == "CH4":
        print("Methane")
    else:
        print("Unknown compound")
```

***** variable *text* is in Python namespace *********

Loops

Looping until a condition is reached or something happens: while

- interrupting flow, maybe when errors: break

```
text = ""
while text != "quit":
    text = input("Please enter a chemical formula H2O,NH3,CH4 (or 'quit' to exit): ")
    if text == "quit":
        print("exiting program...")
        break  # new, exit the while loop - go to next code block below OUTSIDE loop
    elif text == "H2O":
        print("Water")
    elif text == "NH3":
        print("Ammonia")
    elif text == "CH4":
        print("Methane")
    else:
        print("Unknown compound")
```

Loops

Looping until a condition is reached or something happens

```
text = "kjhefkjhe4lkj45jhj43jlkj435444kuj5h34u53h45kj4"
digit_index = -1 # This will be -1 until we find a digit.
# we want to find first digit, then do something else
for i in range(len(text)):
    # If s[i] is a digit
    if text[i].isdigit():
        digit_index = i
        print(digit_index)
```

```
9
13
14
18
19
24
25
```

Loops

Looping until a condition is reached or something happens

```
text = "kjhefkjhe4lkj45jhj43jlkj435444kuj5h34u53h45kj4"
digit_index = -1 # This will be -1 until we find a digit.
# we want to find first digit, then do something else
for i in range(len(text)):
    # If we haven't found a digit, and s[i] is a digit
    if digit_index == -1 and text[i].isdigit():
#or if text[i].isdigit() and digit_index == -1:
        digit_index = i
        print(digit_index)
```

9

- but how many times did it go through the block?

Loops

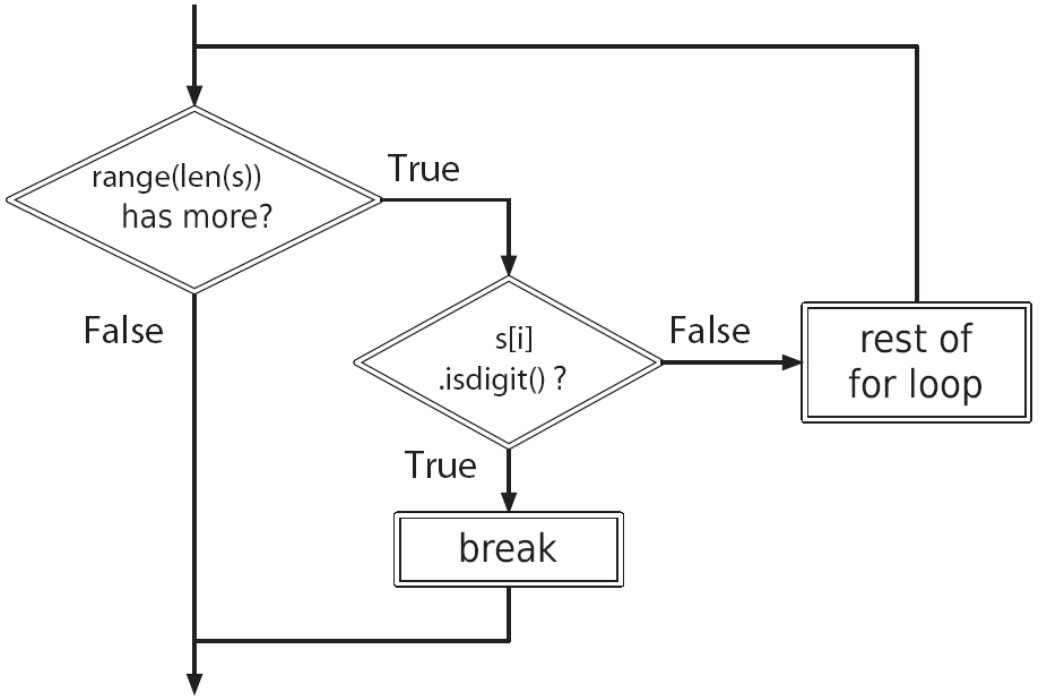
Looping until a condition is reached or something happens

- interrupting flow, maybe when errors: break

```
text = "kjhefkjhe4lkj45jhj43jlkj435444kuj5h34u53h45kj4"
digit_index = -1 # This will be -1 until we find a digit.
# we want to find first digit, then do something else
for i in range(len(text)):
    # If we find a digit
    if text[i].isdigit():
        digit_index = i
        print(digit_index)
        break
```

9

- it stopped after finding the digit.



Loops

Looping until a condition is reached or something happens: while

- cycle back to top of block, keep going, maybe when errors: continue

```
# second try
text = "kjhefkjhe4lkj45jhj43jlkj435444kuj5h34u53h45kj4"
sumDfound = 0 # The sum of the digits seen so far.
numDfound = 0 # The number of digits seen so far.
for i in range(len(text)):
    if text[i].isalpha():
            continue
    sumDfound = sunDfound + int(text[i])
    numDfound = numDfound + 1

print(sumDfound)
print(numDfound)
```

```
77
19
```

```
>>> s = 'C3H7'
>>> total = 0 # The sum of the digits seen so far.
>>> count = 0 # The number of digits seen so far.
>>> for i in range(len(s)):
...     if s[i].isalpha():
...         continue
...     total = total + int(s[i])
...     count = count + 1
...
>>> total
10
>>> count
2
```

Loops

Looping until a condition is reached or something happens: while

- cycle back to top of block, keep going, maybe when errors: continue

```
# even more efficient
text = "kjhefkjhe4lkj45jhj43jlkj435444kuj5h34u53h45kj4"
sumDfound = 0 # The sum of the digits seen so far.
numDfound = 0 # The number of digits seen so far.
for i in range(len(text)):
    if not text[i].isalpha():
        sumDfound += int(text[i])
        numDfound += 1

print(sumDfound)
print(numDfound)
```

Loops

Looping until a condition is reached or something happens: while

- cycle back to top of block, keep going, maybe when errors: continue

```
# super efficient
text = "kjhefkjhe4lkj45jhj43jlkj435444kuj5h34u53h45kj4"
sumDfound = 0 # The sum of the digits seen so far.
numDfound = 0 # The number of digits seen so far.
for i in range(len(text)):
    if text[i].isdigit():
        sumDfound += int(text[i])
        numDfound += 1

print(sumDfound)
print(numDfound)
```

Integration: Doing the right steps in the right order

Loading necessary modules

Setting variables, getting input AND *** knowing what types they should be ***

Write useful functions for special calculations (could save in a Module)

Use methods as necessary (very efficient)

Store (reference) data in lists

if - elif: branch according to choice (user) or condition

while: repeat until something happens

Integration: Chinese Zodiac Program

- similar to what appeared on placemats in Chinese restaurants

Depending on the year you were born, you have certain characteristics

Each year is associated with an animal

The years go in 12 year cycles

Starting in 1900, they are:

Rat, Ox, Tiger, Rabbit, Dragon, Snake, Horse, Goat, Monkey, Rooster, Dog, Pig

http://www.chinahighlights.com/travelguide/chinese-zodiac/ox.htm

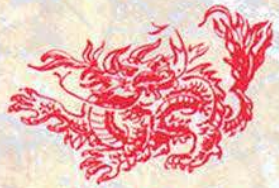
CHINESE ZODIAC

The Chinese Zodiac consists of a 12 year cycle, each year of which is named after a different animal that imparts distinct characteristics to its year. Many Chinese believe that the year of a person's birth is the primary factor in determining that person's personality traits, physical and mental attributes and degree of success and happiness throughout his lifetime. To learn about your Animal Sign, find the year of your birth among the 12 signs running around the border. If born before 1936, add 12 to the year you were born to find your year.

RABBIT

1939, 1951, 1963, 1975, 1987, 1999

Luckiest of all signs, you are also talented and articulate. Affectionate, yet shy, you seek peace throughout your life. Marry a Sheep or Boar. Your opposite is the Cock.



DRAGON

1940, 1952, 1964, 1976, 1988, 2000

You are eccentric and your life complex. You have a very passionate nature and abundant health. Marry a Monkey or Rat late in life. Avoid the Dog.

SNAKE

1941, 1953, 1965, 1977, 1989, 2001

Wise and intense with a tendency towards physical beauty. Vain and high tempered. The Boar is your enemy. The Cock or Ox are your best signs.

HORSE

1942, 1954, 1966, 1978, 1990, 2002

Popular and attractive to the opposite sex. You are often ostentatious and impatient. You need people. Marry a Tiger or a Dog early, but never a Rat.

SHEEP

1943, 1955, 1967, 1979, 1991, 2003

Elegant and creative, you are timid and prefer anonymity. You are most compatible with Boars and Rabbits but never the Ox.

MONKEY

1944, 1956, 1968, 1980, 1992, 2004

You are very intelligent and are able to influence people. An enthusiastic achiever, you are easily discouraged and confused. Avoid Tigers. Seek a Dragon or a Rat.

TIGER

1938, 1950, 1962, 1974, 1986, 1998

Tiger people are aggressive, courageous, candid and sensitive. Look to the Horse and Dog for happiness. Beware of the Monkey.

OX

1937, 1949, 1961, 1973, 1985, 1997

Bright, patient and inspiring to others. You can be happy by yourself, yet make an outstanding parent. Marry a Snake or Cock. The Sheep will bring trouble.

RAT

1936, 1948, 1960, 1972, 1984, 1996

You are ambitious yet honest. Prone to spend freely. Seldom make lasting friendships. Most compatible with Dragons and Monkeys. Least compatible with Horses.

COCK

1945, 1957, 1969, 1981, 1993, 2005

A pioneer in spirit, you are devoted to work and quest after knowledge. You are selfish and eccentric. Rabbits are trouble. Snakes and Oxen are fine.

DOG

1946, 1958, 1970, 1982, 1994, 2006

Loyal and honest you work well with others. Generous yet stubborn and often selfish. Look to the Horse or Tiger. Watch out for Dragons.

BOAR

1947, 1959, 1971, 1983, 1995, 2007

Noble and chivalrous. Your friends will be lifelong, yet you are prone to marital strife. Avoid other Boars. Marry a Rabbit or a Sheep.

The Year of the Ox

The Ox occupies the second position in the Chinese Zodiac. The 12 zodiac animals are, in order: Rat, Ox, Tiger, Rabbit, Dragon, Snake, Horse, Goat, Monkey, Rooster, Dog, and Pig. Each year is related to a Chinese zodiac animal according to the 12-year-cycle.

- **Lucky Colors:** white, yellow, green
- **Lucky Numbers:** 1, 4
- **Lucky Flowers:** tulip, morning glory, peach blossom
- **Year of Birth:** 1913, 1925, 1937, 1949, 1961, 1973, 1985, 1997, 2009, 2021

Ox

The Chinese Zodiac "Ox"

Chinese: 牛 Niú
Years of the Ox: 1913, 1925, 1937, 1949, 1961, 1973, 1985, 1997, 2009, 2021
Personality: diligent, reliable, honest, determined and ambitious
Best-suited careers: agriculture, manufacturing, pharmacy, mechanics, engineering, and real estate

What's Lucky

Lucky numbers	Lucky colors	Lucky flowers	Lucky directions	Lucky lunar months
1 4	white	tulip	N	

What's Lucky

Lucky numbers	Lucky colors	Lucky flowers	Lucky directions	Lucky lunar months
1 4 and numbers containing 1 and 4 (like 14 and 41)	white, yellow, green	tulip, peach blossoms	N S north, south	7th, 9th

Are You an Ox?

Years of the Ox include: 1913, 1925, 1937, 1949, 1961, 1973, 1985, 1997, 2009, and 2021.

You can use the tool on the right to find your zodiac animal sign.

The Oxes' Personality: Diligent, Dependable...

Oxes are known for diligence, dependability, strength and determination. Having an honest nature, Oxes have a strong patriotism for their country, have ideals and ambitions for life, and attach importance to family and work. These things reflect the traditional characteristics of conservatives. Women belonging to the Ox zodiac sign are traditional, faithful wives, who attach great importance to their children's education

You can use the tool on the right to find your zodiac animal sign.

The Oxes' Personality: Diligent, Dependable...

Oxes are known for diligence, dependability, strength and determination. Having an honest nature, Oxes have a strong patriotism for their country, have ideals and ambitions for life, and attach importance to family and work. These things reflect the traditional characteristics of conservatives. Women belonging to the Ox zodiac sign are traditional, faithful wives, who attach great importance to their children's education.

Having a desire to advance and great patience, Oxes can achieve their goals by consistent efforts. They are not influenced by others or the environment, but persist to do things in accordance with their ideas and capabilities. Before taking action, they will have a definite plan with detailed steps and add their strong faith and physical strength. So people of the Ox zodiac sign enjoy great success as a result.

The most disadvantageous trait in Oxes is poor communication skills. They are not good at communicating with others, and even think it not worthwhile exchanging ideas with others. They are stubborn and stick to their own ways.

Good Health for "Oxes"

Oxes are strong and robust; they can enjoy a fairly healthy and long life, fulfilled lives, and little illness.

Because of hard work with a stubborn personality, they often spend too much time in their work, rarely allowing themselves enough time to relax, and tend to forget meals, which make them have intestinal problems. So enough rest and a regular diet are needed for Oxes to work efficiently.

Most Popular Articles on Chinese Zodiac

- 7 Simple Facts on Chinese Zodiac
- 6 Facts on the Year of the Monkey You Should Know
- Lucky Numbers and Colors in Chinese Culture
- How to Bring Yourself Good Luck in Your Zodiac Year

Integration: Chinese Zodiac Program

- similar to what appeared on placemats in Chinese restaurants

Depending on the year you were born, you have certain characteristics

Each year is associated with an animal

The years go in 12 year cycles (12 animals, 12 kinds of people)

We want to tell people what animal they are, and characteristics.

so...

What do we need to do?

Integration: Chinese Zodiac Program

Looping until a condition is reached or something happens: while

```
# Chinese Zodiac Program
import datetime
# initialize animal list
zodiac_animals = ['Rat','Ox','Tiger','Rabbit','Dragon','Snake','Horse', 'Goat','Monkey','Rooster','Dog','Pig']

# characteristics
rat='Forthright , industrious, sensitive, intellectual, sociable'
ox ='Dependable , methodical , modest , born leader, patient'
tiger ='Unpredictable , rebellious, passionate, daring, impulsive'
rabbit ='Good friend, kind, soft-spoken , cautious , artistic'
dragon='Strong, self-assured, proud, decisive, loyal'
snake ='Deep thinker, creative, responsible, calm, purposeful'
horse ='Cheerful, quick-witted, perceptive, talkative, open-minded'
goat ='Sincere, sympathetic, shy, generous, mothering'
monkey ='Motivator, inquisitive, flexible, innovative, problem solver'
rooster ='Organized, self-assured, decisive, perfectionist, zealous'
dog='Honest , unpretentious, idealistic, moralistic, easy going'
pig='Peace-loving, hard-working , trusting, understanding, thoughtful'
```

Integration: Chinese Zodiac Program

Looping until a condition is reached or something happens: while

```
# Chinese Zodiac Program - part 2
characteristics = (rat , ox , tiger, rabbit, dragon, snake , horse , goat , monkey, rooster, dog , pig)
terminate = False

# program greeting
print('This program will display your Chinese zodiac sign and associated')
print('personal characteristics . \n')

# get current year from module datetime
current_yr = datetime.date.today().year

while not terminate :
    # get year of birth
    birth_year = int(input('Enter your year of birth (yyyy) :'))
    # trap birth year errors
    while birth_year < 1900 or birth_year > current_yr :   # perfect for error trapping
        print('Invalid year . Please re- enter\n')
        birth_year = int(input('Enter your year of birth (yyyy) :'))
```

Integration: Chinese Zodiac Program

Looping until a condition is reached or something happens: while

```
# Chinese Zodiac Program - part 3


    # get index of animal based on birth year
    cycle_num = (birth_year - 1900) % 12

    # output results
    print('Your Chinese zodiac sign is the', zodiac_animals[cycle_num] ,'\n')
    print('Your personal characteristics ...')
    print(characteristics [cycle_num] )

    # continue?
    response= input('\nWould you like to enter another year? (y/n) :')
    while response !='y'and response !='n':  # again, perfect for trapping errors
        response= input( "Please enter'y'or'n': ")
    if response == 'n': #    if response=='n': less readable?
        terminate = True
```

Chinese_Zodiac.py

Exam

Some paper (writing), some Python things to do;

open book, open notes, slides, etc.

BUT NO INTERNET USE!

Sample Exam Questions

What is the result of the following statements? (translate into Python code)

```
(3 + 5)^5
```

Are these legitimate statements? If not, why not?

```
gray = '324r2j243'234k234'324l324k'
blue = '324r2j243"234k234'324l324k"
s9 = 93.4455
s10 = 9,4,5,6 # now accepted. What is the Python data type? (know the ones up to now!)
type(s10)
<class 'tuple'>
10s = 'teness'
```

Come up with better variable names:

```
0 = oxygen concentration
r = count of rabbits
pi = percent injured
WTF = waffles, true or false? (Boolean)
```

Sample Exam Questions

What is wrong with the following code? Fix all errors so it will run at least once.

```
rabbits = 0
# stop at 20
while rabbits >= 200
# increment rab
rabbits = rabbits + 1
    when # done print rabbits
    print(rabbits)
```

Write a program to extract all the numerals from this string and call it MyNums

```
'324r2j243"234k234"324l324k'
```

what about:

```
'324r2j243'234k234'324l324k'
```

Sample Exam Questions

What will the following code produce when run:

(program code with loops, ifs, etc.)

```
REFRAIN = '%d bottles of beer on the wall, %d bottles of beer,' + \
'take one down, pass it around, %d bottles of beer on the wall!'

bottles_of_beer = 99
while bottles_of_beer > 1:
    print (REFRAIN % (bottles_of_beer, bottles_of_beer,
        bottles_of_beer - 1) )
    bottles_of_beer -= 1
```

Sample Exam Questions

What will the following code produce when run:

```
rabbits = 0
while rabbits >= 200
rabbits = rabbits + 1
print(rabbits)
```

Improve this code!

```
rabbits = 200
while rabbits <= 3000
    print(i, 'rabbits')
    rabbits = rabbits + 4
```

When do we prefer an *if* loop to a *while* loop? Why?