GIS Programming (4)

GIS 400/500

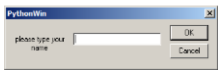
Yanli Zhang · Arthur Temple College of Forestry and Agriculture
Stephen F. Austin State University

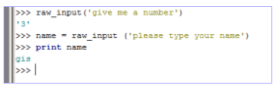
Outline

- Decision making
- Loop

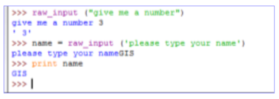
Getting user input

- Use the raw_input method
- It always <u>returns a **string**</u>



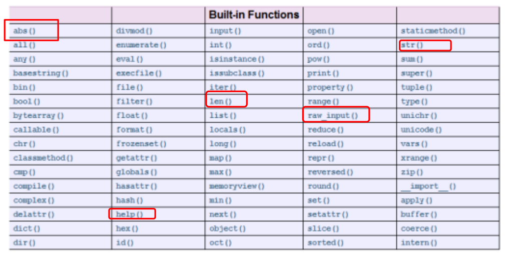


```
>>> raw_input('give me a number')
'3'
>>> name = raw_input ('please type your name')
>>> print name
gis
>>> |
```



```
>>> raw_input ("give me a number")
give me a number 3
' 3'
>>> name = raw_input ('please type your name')
please type your nameGIS
>>> print name
GIS
>>> |
```

Built-in 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()	apply()
delattr()	help()	next()	setattr()	buffer()
dict()	hex()	object()	slice()	coerce()
dir()	id()	oct()	sorted()	intern()

http://docs.python.org/library/functions.html#float

Decision making/branching

- Fundamental part of computer programming
- Make a decision to take one path or another
- Use the *if* structure
  - All *if* have a condition (an expression that is either true or false)
  - *If else*
  - *If elif else*

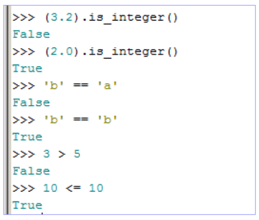
Comparison operators

- Use relational operators (<, >, !=, ==, >=, <=)
  - Return a Boolean result (true/false)
- Use functions
  - TypeName
    - *If TypeOf pLayer is IFeatureLayer*

<	strictly less than
<=	less than or equal
>	strictly greater than
>=	greater than or equal
==	equal
!=	not equal (can be written as <>)
is	object identity
is not	negated object identity

Comparison operators

- String comparison's result is based on alphabetical order

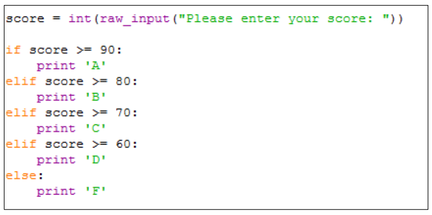


```
>>> (3.2).is_integer()
False
>>> (2.0).is_integer()
True
>>> 'b' == 'a'
False
>>> 'b' == 'b'
True
>>> 3 > 5
False
>>> 10 <= 10
True
```

Code blocks

- A block is one or more consecutive lines indented by the same amount.
- Indenting sets lines off not only visually, but logically too.
- It is required (not optional).
- Put a : (colon) after the condition statement.

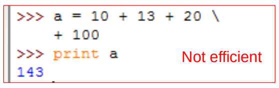
Code blocks example



```
score = int(raw_input("Please enter your score: "))

if score >= 90:
    print 'A'
elif score >= 80:
    print 'B'
elif score >= 70:
    print 'C'
elif score >= 60:
    print 'D'
else:
    print 'F'
```

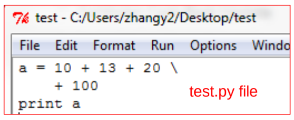
How to handle code blocks?

- Use escape sign: back slash



```
>>> a = 10 + 13 + 20 \
     + 100
>>> print a
143
```

Not efficient

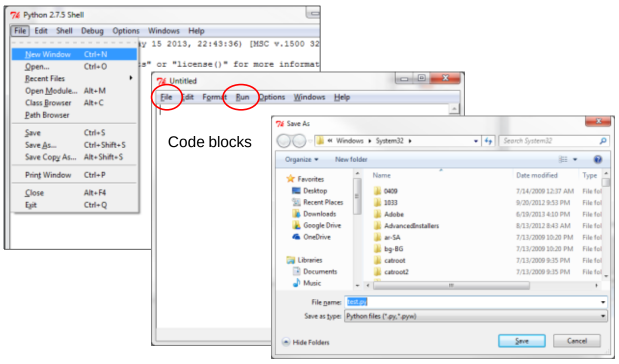


test - C:/Users/zhangy2/Desktop/test

```
a = 10 + 13 + 20 \
    + 100
print a
```

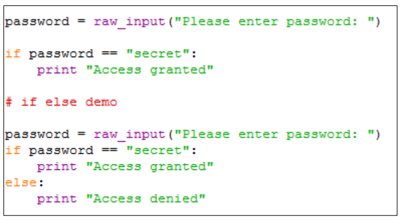
test.py file

.py file



Code blocks

If demo. 1



```
password = raw_input("Please enter password: ")

if password == "secret":
    print "Access granted"

# if else demo

password = raw_input("Please enter password: ")
if password == "secret":
    print "Access granted"
else:
    print "Access denied"
```

If demo. 2

```
score = int(raw_input("Please enter your score: "))

if score >= 90:
    print 'A'
elif score >= 80:
    print 'B'
elif score >= 70:
    print 'C'
elif score >= 60:
    print 'D'
else:
    print 'F'
```

Demo.

- Nested if/else statement

```
if gender == 'Male':
    if age > 12:
        discount = 5
    else:
        discount = 10
else:
    if age > 12:
        discount = 10
    else:
        discount = 15
```

Outline

- Decision making
- Loop

Loop

- Tasks can be run repeatedly with looping statements
- *while* loop
  - It evaluates a logical expression and then decides whether or not to run its block of code.
  - It will run its code until the expression’s true or false status changes.
- *for* loop
  - The loop runs for a specific number of times and the loop ends

While demo.

- The print statement automatically appends a new line to output. To print without a newline, add a **comma** after the last object (*print counter ,)*.

counter = 10 counter >0 True

enter loop, print, counter -1

counter = 9 counter > 0 True

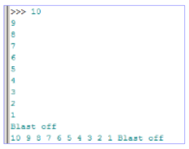
enter loop, print, counter -1

counter = 8 counter > 0 True

enter loop, print, counter -1

……

```
counter = 10

while counter > 0 :
    print counter
    counter = counter - 1

print "Blast off"
```



```
>>> 10
9
8
7
6
5
4
3
2
1
Blast off
10 9 8 7 6 5 4 3 2 1 Blast off
```

Counter

- Counter needs a starting value.
- *Counter needs to be checked in While statement*
  - *Do the job here*
- *Counter needs update within the while block*

```
counter = 10

while counter > 0 :
    print counter
    counter = counter - 1

print "Blast off"
```

Endless loop

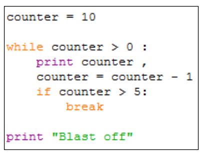
- What will happen?

counter = 9
while counter > 0 :
    print counter,
    counter = counter + 1

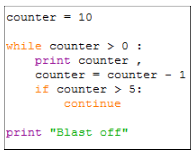
Loop control

- You can use the **break** statement to cause early termination. The **break** statement causes the execution to immediately exit the **while** block and continue with the next statement after the **while** block, if any. (break out of the loop)
- You can use the **continue** statement to skip certain iteration. The **continue** statement causes the execution to skip the rest of the statements within the **while** block for that iteration.(jump back to the top of the loop)

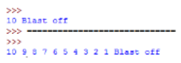
Loop control demo. 1



```
counter = 10

while counter > 0 :
    print counter ,
    counter = counter - 1
    if counter > 5:
        break

print "Blast off"
```

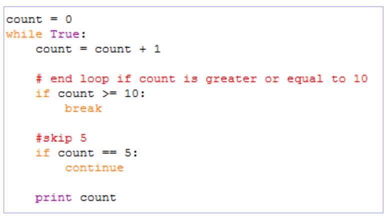


```
counter = 10

while counter > 0 :
    print counter ,
    counter = counter - 1
    if counter > 5:
        continue

print "Blast off"
```

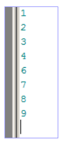


```
>>>
10 Blast off
>>> ------------------------------
>>>
10 9 8 7 6 5 4 3 2 1 Blast off
```

Loop control demo. 2



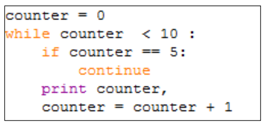
```
count = 0
while True:
    count = count + 1

    # end loop if count is greater or equal to 10
    if count >= 10:
        break

    #skip 5
    if count == 5:
        continue

    print count
```



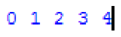
```
1
2
3
4
6
7
8
9
```

Loop control demo. 3

- What will happen?



```
counter = 0
while counter  < 10 :
    if counter == 5:
        continue
    print counter,
    counter = counter + 1
```



```
0 1 2 3 4
```

Does it stop?

Summary of *while*

1. Counter
2. While
3. Check counter
4. Do the job
5. Counter update

6. Loop control may be needed

Code practice

- Give the user 3 times to input password
  - Use *raw_input* to take the input as password
  - Use *if* to make sure the password is correct
  - Use *while* to count input times
  - Use *print* to tell whether the password is correct
  - Any problems?

Hint

```
password = raw_input("Please enter password: ")

if password == "secret":
    print "Access granted"
```

1. Counter
2. While
3. Check counter
4. Do the job
5. Counter update

- If user tried less than 3 times, is loop control needed?

Homework

- How to add 1 to 100 together with loop?
  - 1 + 2 + 3 + 4 + 5 +…. + 100

Summary

- Decision making
  - if
- Loop
  - while
  - Loop control
    - break
    - continue