

Programming Variables

Algebra

$x = 7$ $z = 2(a) + 8$

Python

$x = 7$ (not the same as $X = 7$)
`x = 'dog'` `pet = 'dog'` **TPT#1**
`my_pet = 'rover'` **TPT#2**
`our_pets = ['rover', 'muffy', 'goldie']`

C++

```
int x;  
x = 7;  
x = 'dog'; //Error! Wrong type.
```

Assignment



`X` = 7

Type
Integer



`Pet` = 'dog'

String

Different from $x = = 7$

Weird Behavior: `new_list = [3, x, 'spam']` #change x, x will change everywhere!

Variable types:

integer, float, string, list, dictionary, tuple...
name = reference to object!

BASIC STATEMENTS

Essential power of programming



Do really boring things, over and over again!

By combining Basic Statements you determine what is done by the computer program.

Examples:

- (1) Search a string of 500 bases of a sequence for a particular sequence motif 'GATT'.
- (2) Find the phone number of Smith in a dictionary.

I. *If/elif tests*

```
if <test1>: ← colon
    <statement> ← indentation
elif <test2>:
    <statement>
else:
    <statement>
```

If/elif Examples (see pg. 30 for operators)

```
x = 80
if x >= 0 :
    print "Positive"
else:
    print "Negative"
```

```
y = 99
if y > 0 and y < 100:
    print "between 1 and 100"
elif y > 100:
    print "big"
else:
    print "negative number"
```

```
rabbit = "killer rabbit"
if rabbit == "bugs"
    print "what's up doc?"
elif rabbit == "roger"
    print "how's jessica?"
else:
    print "Run away, run away!"
```

```
z = -1
if z > 0: return 1
else: return 0 ←
```

II. *While loop*

```
while <test>:  
    <statement1>  
else:  
    <statement2>
```

While Examples (see pg. 30 for operators)

```
a, b = 0, 10  
while a < b:  
    print a  
    a = a + 1
```



```
x = y/2  
while x > 1:  
    if y % x == 0:  
        print y, 'has factor', x  
        break  
    x = x - 1  
else: print y, 'is prime'
```



```
food = "spam!"  
while food:  
    print food  
    food = x[1:]
```

```
while 1:  
    print "Bill Gates is evil!"
```

III. *For loop*

for <target> in <object>: <statements>	for <target> in <object>: <statements>
else: <statements>	if <test>: break else: continue
	else: <statements>

For loop Examples (see pg. 30 for operators)

```
food_list = ['eggs', 'bacon', 'cheese']
```

```
for food in food_list:
```

```
    print food
```

```
x = ""
```

```
for food in food_list:
```

```
    x = x + food
```

```
star_trekker = "Cap' Kirk"
```

```
for i in star_trekker:
```

```
    print i
```

```
items = ["a", 1, (3,5)]
```

```
tests = [(3,5), 3.14]
```

```
for x in tests:
```

```
    for item in items:
```

```
        if x == item:
```

```
            print "found ", item
```

```
            break
```

```
        else: print "not found"
```

Breaking Away....

Break, continue, pass & return

break - jumps out of the closest enclosing loop

continue - jumps to top of closes enclosing loop

pass - does nothing at all

return - returns a value ending the loop

IV. The *Range* Function

```
range(1,10)      [1,2,3,4,5,6,7,8,9]
range (5)
range(1,50, 2)
```

```
total = 0
for num in range(5):
    total = total + num
    if total != 10:
        print "not yet!"
    else: break
```

V. Something Useful!

```
file = open("filename", 'r')  OR
while 1:
    line = file.readline()
    if not line: break
```

```
for line in open('filename', 'r').readlines():
    <process line here>
```