Variáveis e Substituição:

```
lista = [ 1, 2, "texto", 3.5 ]
print lista[ 0 ]      # imprime 1
print lista[ 1 : 2 ]  # imprime [ 2, "texto" ]
print lista[ : -1 ]   # imprime [ 1, 2, "texto" ]
lista += [ "novo" ]
print lista           # imprime [ 1, 2, "texto", 3.5, "novo" ]

tupla = ( 1, 2, "texto", 3.5 ) # Elementos não podem ser alterados!
print tupla[ 0 ]      # imprime 1
print tupla[ 1 : 2 ]  # imprime ( 2, "texto" )
print tupla[ : -1 ]   # imprime ( 1, 2, "texto" )
tupla += ( "novo", )
print tupla           # imprime ( 1, 2, "texto", 3.5, "novo" )

dicionario = { "chave": "valor", "c2": "v2" }
print dicionario[ "chave" ] # imprime valor

newstring1 = "string='%s' int='%d' float='%03.2f'" % ( "txt", 12, 4.56 )
newstring2 = "chave=%(chave)s  c2=%(c2)s" % dicionario
newstring3 = "chave=%s  c2=%s" % ( dicionario[ "chave" ],
                                   dicionario[ "c2" ] )
```

Controle de Fluxo e Laços:

```
if a > b and a < c:
    print "a entre b e c"
elif a > c:
    print "a maior que c"
else:
    print "a menor ou igual a b ou igual a c"

for elemento in lista:
    print "elemento: %s" % elemento

coordenadas = [ ( 0, 0, 0 ), ( 1, 0, 0 ), ( 0, 1, 0 ), ( 0, 0, 1 ) ]
for x, y, z in coordenadas:
    print "Ponto: x=%d, y=%d, z=%d" % ( x, y, z )

loop = 1
while loop:
    resultado = faca_acao()
    if resultado < 0:
       break    # Pára o laço
    else resultado > 0:
       continue # Volta para o começo do laço
    print "teste"
```

Funções:

```
def funcao( p1, p2="Valor Padrao" ):
    print "p1: '%s'  p2: '%s'" % ( p1, p2 )

def f_param_variaveis( p1, *args ):
    print "p1: '%s'" % p1
    for arg in args:
        print "   arg: %s" % arg

def f_param_nome_variaveis( p1, **args ):
    print "p1: '%s'" % p1
    for p_name, p_value in args:
        print "   arg: %s=%s" % ( p_name, p_value )
```

Classes:

```
class A:
    attributo = 1
    __privado = 123
    def __init__( self, valor ):
        self.attributo = valor
        self.__metodo_privado()
    def __metodo_privado( self ):
        print "chamando metodo privado"
class B:
    attributo = 2
    def __init__( self ):
        self.novo_attributo = 2
class C( A, B ):
    def __init__( self ):
        B.__init__( self )
class D( B, A ):
    def __init__( self ):
        B.__init__( self )
a = A( 1 )
b = B()
c = C()
d = D()
print a.attributo   # imprime 1
print b.attributo   # imprime 2
print c.attributo   # imprime 1: herança múlt. (A,B) A sobrepõe-se a B
print d.attributo   # imprime 2: herança múlt. (B,A) B sobrepõe-se a A
```

Módulos e Espaço de Nomes:

```
import urllib
url      = "http://www.unicamp.br/" + urllib.quote( "index.html" )
conteudo = urllib.urlopen( url ).read()

# importa símbolos para espaço de nomes atual:
from urllib import *
url      = "http://www.unicamp.br/" + quote( "index.html" )
conteudo = urlopen( url ).read()
```