If-elif-else statement

```
#Grade A:[90,100) marks
#Grade B:[75,90) marks
#Grade C:[50,75) marks
#Grade F:[0,50) marks
marks=int(input('Enter marks: '))
if(marks>=90):
    print ("Grade A")
elif(marks>=75):
    print ("Grade B")
elif(marks>=50):
    print ("Grade C")
else:
    print ("Grade F")
```

Output

```
Enter marks: 88
Grade B
```

While loops

```
Choice=1
while(Choice == 1):
    marks=int(input('Enter marks: '))
    if(marks>=50):
        print ("Student passed the subject")
    else:
        print ("Student failed the subject")
    Choice=int(input('Press 1 to find result of student: '))
```

Output

```
Enter marks: 15
Student failed the subject
Press 1 to find result of student: 1
Enter marks: 93
```

```
Student passed the subject
Press 1 to find result of student: 0
```

For loops

```
nums = [0,59,2,47,1,47,89,656,23,12]
for num in nums:
    print(num)
```

Output

```
0
59
2
47
1
47
89
656
23
12
```

Break statement

```
for n in range(2, 10):
    for x in range(2, n):
        if n % x == 0:
            print(n, 'equals', x, '*', n//x)
            break
        else:
without finding a
            print(n, 'is a prime number')
```

Output

```
2 is a prime number
3 is a prime number
```

```
4 equals 2 * 2
5 is a prime number
6 equals 2 * 3
7 is a prime number
8 equals 2 * 4
9 equals 3 * 3
```

Continue statement

```
for num in range(2, 10):
    if num % 2 == 0:
        print("Found an even number", num)
        Continue
    print("Found a number", num)
```

Output

```
Found an even number 2
Found a number 3
Found an even number 4
Found a number 5
Found an even number 6
Found a number 7
Found an even number 8
Found a number 9
```

Pass statement

```
for letter in 'Python':
    if letter == 'h':
        pass
```

```
        print ('This is pass block')
    print ('Current Letter :', letter)
print ("Good bye!")
```

Output

```
Current Letter : P
Current Letter : y
Current Letter : t
This is pass block
Current Letter : h
Current Letter : o
Current Letter : n
Good bye!
```

FUNCTIONS

```
def my_first_function(): #function head
        print "This is my first Python function!"
#body
```

Generator Functions

Here is a simple example of a generator function which returns 7 random integers:

```
import random
def lottery():
    # returns 6 numbers between 1 and 40
    for i in xrange(6):
```

```
        yield random.randint(1, 40)
    # returns a 7th number between 1 and 15
    yield random.randint(1,15)
for random_number in lottery():
    print "And the next number is... %d!" %
random_number
```

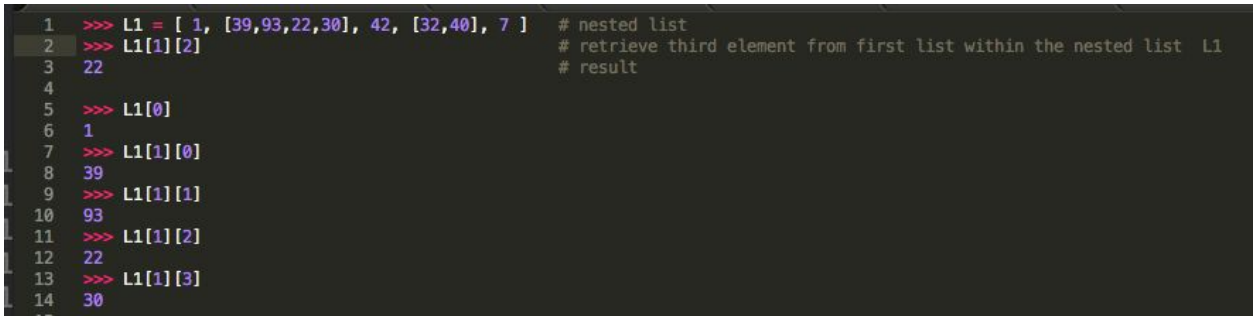
Lists

```
>>> l1 = ['apples', 'bananas', 'melons']
>>> l1
['apples', 'bananas', 'melons']

>>> l2 = list(l1)
>>> l2
['apples', 'bananas', 'melons']

>>> len(l1)
3
```

Nested Lists



```
>>> L1 = [ 1, [39,93,22,30], 42, [32,40], 7 ]   # nested list
>>> L1[1][2]                                    # retrieve third element from first list within the nested list  L1
22                                              # result

>>> L1[0]
1
>>> L1[1][0]
39
>>> L1[1][1]
93
>>> L1[1][2]
22
>>> L1[1][3]
30
```

Lists as Stacks

```
>>> stack = [1,2,3,4,5]
>>> stack.append(6)
>>> stack.append(7)
>>> stack
[1, 2, 3, 4, 5, 6, 7]

>>> stack.pop()
7
>>> stack
[1, 2, 3, 4, 5, 6]

>>> stack.pop()
6

>>> stack.pop()
5

>>> stack
[1, 2, 3, 4]
```

Lists as Queues

```
>>> from collections import deque
>>> queue = deque(["Lisa", "Nicole", "Alex"])
>>> queue.append("Ahmed")        # Ahmed arrives
>>> queue.append("Julio")        # Julio arrives
>>> queue.popleft()              # First person leaves
'Lisa'
>>> queue.popleft()              # Second person leaves too
'Nicole'
>>> queue                        # Three persons are left alone
deque(['Alex', 'Ahmed', 'Julio'])
```

Tuples and sequences

```
>>> cars=("porsche","mercedes","bmw")          # create a tuple "cars"
>>> us_cars=("corvette","chrysler",cars)       # create a second tuple "us_cars"
>>> print  cars[1]                             # retrieve from cars the second element
mercedes
>>> print  us_cars[0]                          # retrieve from us_cars the first element
corvette
>>> print us_cars[2][2]                        # retrieve the third element of "cars" (tuple within a tuple)
bmw
```

```
1   cars=["porsche","BMW","Mercedes Benz"]
2   #indexing
3   print cars[0]
4   porsche
5
6   print cars[1]
7   BMW
8
9   print cars[2]
10  Mercedes Benz
11
12
13  #slicing using a list
14  print cars[0:2]
15  ['porsche', 'BMW']
16
17  print cars[:]
18  ['porsche', 'BMW', 'Mercedes Benz']
19
20
21  #slicing using a string
22  name="peter"
23  print name[0:1]
24  p
25
26  print name[1:4]
27  ete
28
```

Sets

```
>>>en=set("hello")
>>>es=set("hola")
>>>en
set(['h', 'e', 'l', 'o'])
>>>es
set(['a', 'h', 'l', 'o'])


#difference of en & es
>>>en - es
set(['e'])


#union of a & b
>>>en | es
set(['a', 'e', 'h', 'l', 'o'])


#intersection of a & b
>>>a & b
set(['h', 'l', 'o'])

```

Dictionaries

```
#creating a dictionary
>>>season={"winter":"very cold","summer":"hot","autumn":"rainy"}

#add a new item
>>>season["spring"]="nice"

#delete an existing item
>>>del season["winter"]
>>>for i,j in season.items():
... print "in %s it's %s"%(i,j)

"in autumn it's rainy"
"in summer it's hot"
"in winter it's very cold"
"in spring it's nice"



#search for an item
>>>if  season.has_key("summer"):
...     print "This summer will be %s" %season["summer"]
"This summer will be hot"
```

Classes

Getting started

```
class Book ():
     pass
```

Adding detail

```
class Book ():
     def openBook(self):
          print('Open the book.')
     def closeBook(self):
          print('Close the book.')

aBook = Book()
aBook.openBook()
aBook.closeBook()
```

Run the program

```
Open the book.
Close the book.
```

Attributes

Instance variables instead of arguments

```
class Book () :
     def __init__(self, title, author) :
          self.bookTitle = title
          self.bookAuthor = author

     def openBook(self) :
          print('Open the book', self.bookTitle,
```

```
'written by', self.bookAuthor, '.')
     def closeBook(self):
          print('Close the book', self.bookTitle,
'written by', self.bookAuthor, '.')
```

The rest of the program

```
aBook = Book ('Calculus Today', 'Dr. Diffy Q')

aBook.openBook()
aBook.closeBook()
```

Run the program

```
Open the book Calculus Today written by Dr. Diffy
Q .
Close the book Calculus Today written by Dr.
Diffy Q .
```

Inheritance

Multiple Inheritance

```
Class InventorySystem():
     Pass

class Book ():
    def __init__(self, title, author):
        self.bookTitle   = title
        self.bookAuthor  = author
    def openBook(self):
        print('Open the book', self.bookTitle, 'written by',
self.bookAuthor, '.')

class ChildrensBook(InventorySystem, Book):
```

```
    def __init__(self, title, author):
        super().__init__(title, author)
        self.bookPrice = 2.0
    def openBook(self):
        print('Open the childrens book', self.bookTitle, 'written
by', self.bookAuthor, '.')
    def readBookAloud(self):
        print('Read the childrens book', self.bookTitle, 'written
by', self.bookAuthor, 'aloud.')
```

Recursion

```
# An example of a recursive function to
# find the factorial of a number
def fact(x):
    """This is a recursive function
    to find the factorial of an integer"""
    if x <= 1:
        return 1
    else:
        return (x * fact(x-1))
num = int(input("Enter a number: "))
if num >= 1:
    print("The factorial of", num, "is", fact(num))
```

Syntax Errors

Misspelling Python keywords

```
1    counter=0
2    While counter < 5:
3        print "hello"
```

```
4        counter = counter + 1

Traceback (most recent call last):
   File "./tester.py", line 2
    While counter < 5:
                ^
SyntaxError: invalid syntax
```

Missing off a colon from the end of an `if` or `while` line

```
1    counter = 0
2    if counter == 4
3        print "counter is 4"

  File "tester.py", line 2
    if counter == 4
                  ^
SyntaxError: invalid syntax
```

Wrong number of brackets in function calls or expressions

```
1    x = 3
2    y = ((x + 3) * (x + 4)
3    print y

  File "tester.py", line 3
    print y
        ^
SyntaxError: invalid syntax
```

Incorrect indentation

```
1         x = 0
2     y = 1

  File "tester.py", line 1
    x = 0
    ^
SyntaxError: invalid syntax
```

Runtime Errors

The Name Error will give us a Traceback message like this:

```
Traceback (most recent call last):
  File
"C:/Users/John/Documents/Teaching-BU/Python-debugging/test.py",
line 7, in
    main()
  File
"C:/Users/John/Documents/Teaching-BU/Python-debugging/test.py",
line 5, in main
    print hello
NameError: global name 'hello' is not defined
```

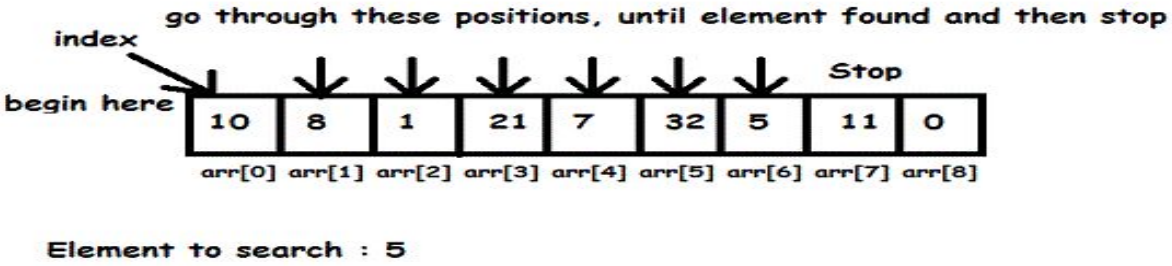
A TypeError you might encounter may look like this:

```
File
"C:/Users/John/Documents/Teaching-BU/Python-debugging/test.py",
line 2, in
    print "I am %d feet %d inches tall" % (5, 2, 5)
TypeError: not all arguments converted during string formatting
```

Or:

```
File
"C:/Users/John/Documents/Teaching-BU/Python-debugging/test.py",
line 2, in
    print "I am %d feet %d inches tall" % (5)
TypeError: not enough arguments for format string
```

Linear Search



In the above example.We will first set I =0 and found to false. Then we will compare it with each element until we reach the end.If we find our desired element in between then we will stop set found=true and return the value of I and found.In this case, we will return (6,True).That means the element is found at 6th index.

Binary Search

Initially,set start=0, end=11 ,found=false

Loop 1:

mid=(0+11)/2=5(truncated).Since, 19>8 set start=5+1=6.

Loop 2:

mid=(6+11)/2=8.Since, 19<21 set end=8-1=7.

Loop 3:

mid=(6+7)/2=6.Since, 19>11 set start=6+1=7.

Loop 4:

mid=(7+7)/2=7.Since 19=19 set found=true and Return mid,found//Output:(7,True)