Lecture 5: Control Flows and Functions

Control Flows

In a typical programming language, the major control flows include **Choice** and **Loop**.

Choice and `if` loops in Python

General form:

```
if test_1:     # test_1 should return a boolean result -- don't forget the colon: here
    statement_1 # associated block of test_1 -- don't forget the indentation here
elif test_2:   # optional, if we have multiple branches
    statement_2
else:          # optional
    statement_3
```

In [ ]:
```
x = -5

if x > 0:
    print('positive number')
elif x == 0:  # using == to test the equivalence of values
    print('zero')
else:
    print('negative number')
```

In [ ]:
```
x = 1
mylist = [1,2,3]

if x in mylist: # using keyword "in" to test if x is the element of list
    print('x is in the list')
else:
    print('x is not in the list')
```

In [ ]:
```
x = 10
if x > 0 or x < 0: ##  "and,or,not" are three typical boolean expressions in python
    print('non-zero number')
else:
    print('zero number')
```

In [ ]:
```
x = 10
if not x == 0: # or you can write if x!=0
    print('non-zero number')
else:
    print('zero number')
```

Loop: `while`

```
while test:  # test returns a boolean
    statement_1
else:             # a special feature about python that is overlooked! Use it in combination
with break/continue
    statement_2
```

In [ ]:
```
n = 0
mylist = [] # create an empty list
while n < 10:
    mylist.append(n) # the code to be executed if n < 10
    n = n + 1 # increase the counter by 1
    print(id(mylist))

print(mylist) # this line is no longer in the while loop!
```

In [ ]:
```
# determine whether y is prime
y = 3
x = y // 2 # Why? Can it be improved?
while x > 1:
    if y % x == 0: # Reminder
        print('y is not prime')
        break      # exit the while loop immediately
    else:          # this else is for if
        x = x-1
else:              # this else is for while -- run this if only there is normal exit
 without hitting the break
    print('y is prime') # what if this statement is not in the else block?

print(x)
```

Loop: `for`

```
for target in object:
    statement_1
    if test_1: break # exit the for loop immediately
else:                # run this only when exit normally without hitting break
    statement_2
```

Computing sum of the list

- Iterating the list directly
- Iterating through the index

In [ ]:
```
#iterating the list
mylist = [1,2,3,4]
mysum = 0

for x in mylist:
    mysum = mysum + x
print(mysum)

# this might be a more pythonic way!
```

In [ ]:
```
#iterating through index
mylist = [1,2,3,4]
mysum = 0

for i in range(len(mylist)):
    mysum = mysum + mylist[i]
print(mysum)

# this is what you're familiar in Matlab perhaps!
```

By using the `enumerate()` we can actually iterate in both ways simultaneously!

In [ ]:
```
mylist = [[1,2],[3,4]]

for (i,x) in enumerate(mylist): # pay attention to the order (i,x)
    print(x)
    print(id(x))
    print(mylist[i])
    print(id(mylist[i]))
```

Change the elements of list

In [ ]:
```
mylist = [1,2,3,4]
print(id(mylist))

for i in range(len(mylist)):
    mylist[i] = mylist[i] + 1

print(mylist)
print(id(mylist))
```

In [ ]:
```
# this will NOT work -- think why !
mylist = [1,2,3,4]

for x in mylist:
    x = x + 1

print(mylist)
```

A more *pythonic* way is through list comprehension

```
new_list = [A for B in C if D]
```

In [ ]:
```
mylist = [1,2,3,4]
print(id(mylist))

mylist = [x+1 for x in mylist]

print(mylist)
print(id(mylist))
```

comprehension is very powerful -- it can also be combined with if statement

In [ ]:
```
# take all the special attributes/names of mylist
dir_mylist = dir(mylist)
special_names = [name for name in dir_mylist if name.startswith('__')]
print(special_names)
```

I highly recommend [this video (https://www.youtube.com/watch?v=OSGv2VnC0go)](https://www.youtube.com/watch?v=OSGv2VnC0go) for writing the pythonic codes.

In [ ]:
```
obj = ["Even" if i%2==0 else "Odd" for i in range(100)]
print(obj)
```

In [ ]:
```
vec = [[1,2,3], [4,5,6], [7,8,9]]
vec_flat = [num for elem in vec for num in elem]
print(vec_flat)
```