2. Control Flow

Spring 2018

Control Flow

- 프로그램의 흐름을 제어
  1. 분기문
     - if / elif / else
  2. 반복문
     - for / while
  3. 함수
     - def

- 코드 블록 (Code Block)
  1. 분기, 반복, 함수로 구분되는 코드 단위
  2. 예를 들어, 반복문에서 반복되는 코드들을 구분하는 구역
  3. 파이썬의 코드블록은 줄앞의 공백(indent) 으로 구분함
     - 블록이 아닌데 공백을 넣으면 오류가 발생함
  4. 일반적으로 space 4개를 줄 앞에 넣음

if / elif / else

- a가 짝수이면 “even”을 홀수면 “odd”를 출력

```
if a%2==0:
    print(“even”)
else:
    print(“odd”)
```

- x가 0보다 크면 “positive”, 0보다 작으면 “negative”, 0이면 “zero” 출력

```
if x>0:
    print(“positive”)
elif x<0:
    print(“negative”)
else:
    print(“zero”)
```

if, elif, else 문장 마지막에는 반드시 : (콜론)을 넣음
(for/while/def 에서도 마찬가지임)

ternary expression

- 다음을 간단하게 쓰고 싶다면

type = “call”

if type==“call”:

  d = 1

else:

  d = -1

d = 1 if type==“call” else -1

```
value = true-expr if condition else false-expr
```

```
if condition:
    value = true-expr
else:
    value = false-expr
```

for

- for x in x_values:

  (do something)

```
kinds = [“Gold”, “Silver”, “Bronze”]
for x in kinds:
    print(x + “ Medal”)
```

x는 kinds의 원소인 “Gold”, “Silver”, “Bronze”로 바뀌면서 반복 수행

```
s, n = 0, [1,2,3,4,5,6,7,8,9,10]
for i in n:
    s += i**2
print(“sum of squares = %d” % s)
```

1 부터 10까지 정수를 제곱하여 s 변수에 누적으로 더함

range

- 0 부터 n-1 까지 n개의 연속된 숫자를 쉽게 만들기
  - range(n)
  - [0,1,2, ... ,9]의 리스트로 변환하려면 list(range(10))

- range(start, end, step)

```
s = 0
for i in range(1,11):
    s += i**2
print("sum of squares = %d" % s)
```

```
kinds = ["Gold", "Silver", "Bronze"]
for i in range(len(kinds)):
    print(kinds[i] + " Medal")
```

실행 결과는 동일하지만 추천하지 않음

파이썬은 간결함을 추구하는 언어.

enumerate

```
kinds = [“Gold”, “Silver”, “Bronze”]
for i in range(len(kinds)):
    print(“%d: %s” % (i, kinds[i] + “ Medal”))
```

이 프로그램은 파이썬답게 구현하면,

```
kinds = [“Gold”, “Silver”, “Bronze”]
for i, x in enumerate(kinds):
    print(“%d: %s” % (i, x + “ Medal”))
```

Comprehension

- 앞의 예제에서 kinds 변수를 가지고 [“Gold Medal”, “Silver Medal”, “Bronze Medal”] 의 새로운 리스트를 만드려면?

```
m, kinds = [], [“Gold”, “Silver”, “Bronze”]
for x in kinds:
    m.append(x + “ Medal”)
```

```
kinds = [“Gold”, “Silver”, “Bronze”]
m = [x + “ Medal” for x in kinds]
print(m)
```

while

- 특정 조건이 True인 경우 반복

```
s, c = 0, 0
while s<100:
    c += 1
    s += c
print(c, s)
```

break / continue

- break: loop를 탈출

```
s, c = 0, 0
while True:
    c += 1
    s += c
    if s>100: break
print(c, s)
```

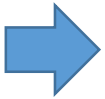
- continue: loop블록 마지막으로 넘어감

```
s, c = 0, 0
while s<100:
    c += 1
    if c%2==0: continue
    s += c
print(c, s)
```

def

- 동일한 코드를 반복적으로 사용해야 한다면 반드시 Function 으로 만들어서 사용해라.
- 동일한 코드를 중복해서 copy & paste하는 일은 최대한 자제해라.

```
from math import *
x = 1
f = 1/sqrt(2*pi)*exp(-x**2/2)
y = 2
g = 1/sqrt(2*pi)*exp(-y**2/2)
z = 3
h = 1/sqrt(2*pi)*exp(-z**2/2)
```



```
from math import *
def pdf(x):
    return 1/sqrt(2*pi)*exp(-x**2/2)

x = 1
f = pdf(x)
y = 2
g = pdf(y)
z = 3
h = pdf(z)
```

default argument

- 함수의 인자 중에서 입력하지 않으면 default 값을 가지도록 함

```
def foo(x, y, z = 1):
    return x*y*z

print(foo(1,2,3))
print(foo(1,2))
```

함수의 return

- 함수는 return 값이 없어도 됨
- return 값으로 list, tuple, dict 모두 가능
    - 여러 값의 결과를 return할 때 사용

Exercise

- 1m 길이의 밧줄을 임의의 길이로 3등분 했을 때 얻어지는 3개의 조각으로 삼각형이 만들어질 확률은 얼마인지 계산하시오.
  1. 삼각형이 만들어지기 위해서는 짧은 두 변의 길이의 합이 가장 긴 변의 길이 보다 길어야한다.
  2. random 모듈의 uniform분포 함수를 이용해서 임의의 두 점 (x,y)을 n차례 생성하고 그 중에서 삼각형이 만들어지는 비율을 계산한다.
  3. n의 횟수를 증가시키면 비율은 정확한 확률에 수렴한다.

- n의 값을 500회, 1000회, ..., 10000회로 변경하여 총 20번의 확률을 계산하시오.

변수 swap: 두 변수의 값을 서로 바꾸는 것

```
x = 10
y = 20
x, y  = y, x
```