Sankify Python 3 課程 Lesson 8 說明

Script	done
Video	

Functions and recursion

> https://snakify.org/en/lessons/functions/

Functions

```
a = 3
for b in range(1,10):
    print(f"{a} x {b} = {a*b:2}")
print()

a = 5
for b in range(1,10):
    print(f"{a} x {b} = {a*b:2}")
print()
```

```
def mta(a):
    for b in range(1,10):
        print(f"{a} x {b} = {a*b:2}")
    print()

mta(3)
mta(5)
```

```
3 x 1 =  3
3 x 2 =  6
3 x 3 =  9
3 x 4 = 12
3 x 5 = 15
3 x 6 = 18
3 x 7 = 21
3 x 8 = 24
3 x 9 = 27

5 x 1 =  5
5 x 2 = 10
5 x 3 = 15
5 x 4 = 20
5 x 5 = 25
5 x 6 = 30
5 x 7 = 35
5 x 8 = 40
5 x 9 = 45
```

Return result

```
def grade(n):
    if n>=90:
        return 'A'
    elif n>=80:
        return 'B'
    elif n>=70:
        return 'C'
    elif n>=60:
        return 'D'
    else:
        return 'F'

from random import randrange

for g in [randrange(30,101) for _ in range(10)]:
    print(f"grade({g:3}) = {grade(g)}")
```

```
grade( 96) = A
grade( 72) = C
grade( 81) = B
grade( 78) = C
grade( 66) = D
grade( 72) = C
grade( 63) = D
grade( 57) = F
grade( 73) = C
grade( 81) = B
```

Cascaded function calls

```
def min3(a, b, c):
```

```
    if b > a < c:
        return a
    elif a > b < c:
        return b
    else:
        return c

from random import randrange

for loop in range(5):
    x, y, z = randrange(100), randrange(100), randrange(100)
    print(f"min3({x},{y},{z}) = {min3(x,y,z)}")
```

```
min3(96,77,20) = 20
min3(22,17,21) = 17
min3(32,48,31) = 31
min3(77,7,17) = 7
min3(4,57,30) = 4
```

```
def min2(a, b):
    if a < b:
        return a
    else:
        return b

def min3(a, b, c):
    return min2(min2(a,b),c)

from random import randrange

for loop in range(5):
    x, y, z = randrange(100), randrange(100), randrange(100)
    print(f"min3({x},{y},{z}) = {min3(x,y,z)}")
```

```
min3(60,75,20) = 20
min3(58,94,30) = 30
min3(49,29,40) = 29
min3(59,91,98) = 59
min3(68,95,84) = 68
```

Accept various number of arguments

```
def minn(*n):
    result = n[0]
    for x in n[1:]:
        if x < result:
            result = x
    return result

print(f"minn(12,34,2,45,7) = {minn(12,34,2,45,7)}")
print(f"minn(12,34,45,7)   = {minn(12,34,45,7)}")
print(f"minn(12,34,45)     = {minn(12,34,45)}")
print(f"minn(34,45)        = {minn(34,45)}")
```

```
minn(12,34,2,45,7) = 2
minn(12,34,45,7)   = 7
minn(12,34,45)     = 12
minn(34,45)        = 34
```

Local and global variables

> https://snakify.org/en/lessons/functions/#section_2

```
def f():
    print(a)

a = 1
f()
```

```
def f():
    a = 1

f()
print(a)
```

```
def f():
    print(a)

a = 1
f()
```

```
1
```

```
def f():
    a = 1

f()
print(a)
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
<ipython-input-1-a6432c295a68> in <module>
      3
      4 f()
----> 5 print(a)

NameError: name 'a' is not defined
```

Local vs. Global variables implicit

The Python interpreter considers `a variable local to the function, if in the code of this function there is at least one instruction that modifies the value of the variable.` Then that variable also cannot be used prior to initialization. Instructions that modify the value of a variable — the operators `=`, `+=`, and usage of the variable as a loop `for` parameter. However, `even if the changing-variable statement is never` the interpreter

```
def f():
    a = 1
    print(a)

a = 0
f()
print(a)
```

```
def f():
    a = 1
    print(a)

a = 0
f()
print(a)
```

```
1
0
```

`executed,` the interpreter cannot check it, and the variable is still local.

```
def f():
    print(a)
    if False:
        a = 0

a = 1
f()
```

```
---------------------------------------
---------------------------------------
UnboundLocalError
Traceback (most recent call last)
<ipython-input-2-bfac009ebf5d> in <mod
ule>
      5
      6 a = 1
----> 7 f()

<ipython-input-2-bfac009ebf5d> in f()
      1 def f():
----> 2     print(a)
      3     if False:
      4         a = 0
      5

UnboundLocalError: local variable 'a'
referenced before assignment
```

Explicit use of global variables

```
def f():
    global a
    a = 1
    print(a)

a = 0
f()
print(a)
```

```
def f():
    global a
    a = 1
    print(a)

a = 0
f()
print(a)
```

```
1
1
```

Return multiple values

```
from random import randrange

def next4Queen(x, y, dist):
    rn = randrange(4)
    d = randrange(-dist, dist+1)
    if rn==0:
```

```
        return x+d, y
    elif rn==1:
        return x, y+d
    elif rn==2:
        return x+d, y+d
    return x+d, y-d

def main():
    for _ in range(10):
        x, y = randrange(1,9), randrange(1,9)
        x1, y1 = next4Queen(x, y, 3)
        print(f"({x}, {y}) ({x1}, {y1}) ", end="")
        print("ckecked" if chk(x,y,x1,y1) else "fail")

def chk(x1, y1, x2, y2):
    c1 = x1==x2 or y1==y2
    c2 = abs(x1-x2) == abs(y1-y2)
    return c1 or c2

if __name__=="__main__":
    main()
```

```
(7, 2) (7, 1) ckecked
(7, 6) (5, 6) ckecked
(6, 5) (3, 8) ckecked
(1, 5) (1, 7) ckecked
(5, 4) (7, 2) ckecked
(7, 6) (5, 8) ckecked
(8, 7) (8, 7) ckecked
(4, 5) (3, 4) ckecked
(5, 6) (4, 6) ckecked
(6, 8) (3, 8) ckecked
```

Out of range!

```
(2, 7) (-1, 10) ckecked
(5, 4) (8, 7) ckecked
(7, 4) (7, 4) ckecked
(6, 6) (6, 4) ckecked
(3, 8) (6, 5) ckecked
(6, 7) (9, 10) ckecked
(7, 3) (7, 3) ckecked
(8, 4) (8, 6) ckecked
(1, 6) (-1, 6) ckecked
(4, 4) (4, 4) ckecked
```

Recursion

```
def sum(n):
    if n>0:
        return n + sum(n-1)
    else:
        return 0

print(f"sum(10) = {sum(10)}")
print(f"sum(100) = {sum(100)}")
print(f"sum(500) = {sum(500)}")
```

```
sum(10) = 55
sum(100) = 5050
sum(500) = 125250
```

Fibonacci Sequence

```
fib(0) = 0
fib(1) = 1

fib(n) = fib(n-2) + fib(n-1)  where n>1
```

```
def fib(n):
    if n<=1:
        return n
```

```
0 1 1 2 3 5 8 13 21 34
```

```
    else:
        return fib(n-2) + fib(n-1)

for i in range(10):
    print(fib(i), end=' ')
```

Snakify 範例解析 - Negative exponent

Given a positive real number a and **integer** n. Compute a**n. Write a function `power(a, n)` to calculate the results using the function and print the result of the expression. `Don't use the same function from the standard library.`

```
def power(a,n):
    p = 1
    for _ in range(abs(n)):
        p*=a
    return p if n>0 else 1/p

a = float(input())
n = int(input())

print(power(a,n))
```

```
2
-3
0.125
```

Snakify 範例解析 - Reverse the sequence

Given a sequence of integers that end with a 0. Print the sequence in reverse order. `Don't use lists or other data structures.` Use the *force* of `recursion` instead.

```
def reverse(n):
    if n!=0:
        reverse(int(input()))
    print(n)

reverse(int(input()))
```

```
1
2
3
0
0
3
2
1
```


該休息了…