

서울대 영재원 코딩수학 탐구자료 및 과제

[http://www.javamath.com/javamaldown.html \(<http://www.javamath.com/javamaldown.html>\)](http://www.javamath.com/javamaldown.html)

먼저 위를 따라 WinPython36F 설치후 탐구 !!!

```
In [ ]: 
```

```
In [ ]: 
```

```
In [ ]: !pip install matplotlib
```

아래 실행때 no module matplotlib 라는 error 가 나오면

!pip install matplotlib 명령으로 라이브러리를 설치 !!

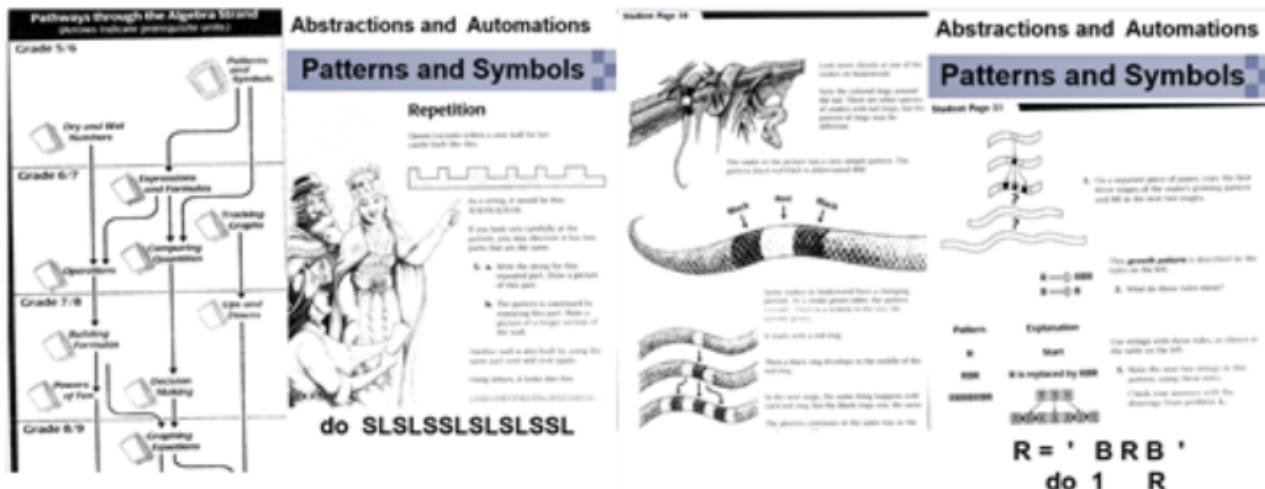
```
In [ ]: 
```

```
In [ ]: 
```

코딩교육의 목표는 컴퓨팅 사고력 (computational thinking) 역량을 키우는 것이다. 컴퓨팅 사고력이란 용어는 거북명령 프로그램 LOGO 를 만들고 (제자들이 마우스 버전으로 만든 것이 고양이가 나오는 스크래치) 1969년에 XOR 문제 풀이로 인공지능 연구에 거울이 오도록 한 MIT 대학의 수학자, 인공지능, 코딩교육자 패럴트 Papert 의 마인드스톱 (1980년 출판) 책이다. 컴퓨팅 사고력의 핵심은 인간의 생각을 컴퓨터 기계가 이해하고 풀 수 있도록 문제를 표현하고 (Abstraction) 자동으로 풀도록 (Automation) 코딩하는 것이다. 다음에 소개하는 진화 알고리즘은 1975년 헐랜드에 의해 제안되었는데, 이전에 1968년에 식물학자 린덴마이어에 의해 식물이 자라는 유전자와 분화규칙으로 고등학교 수학에도 나오는 초기값과 관계식이 주어지는 리컬전 점화수열의 문제로 식물의 성장모델을 제시하였다. 여기에 패럴트의 거북명령 (스크래치의 명령) 앞으로 가지와 옆으로 돌자인 가지와 돌자 명령이 사용되고, 성장의 규칙이 치환문자로 주어진다. 이러한 핵심적인 내용은 1998년에 초등학교 5학년 수학책으로 개발된 프로이엔탈의 RME 수학교과서에도 unplug 언플리그 코딩으로 나와있으며 서울대 수리정보 영재센터 문제에도 나왔다. 1968년에는 인간이 최초로 달에 착륙하기도 하였다. 코딩교육의 목표인 컴퓨팅 사고력 역량을 목표로 하는 서울대 수리정보 영재분야의 탐구는 abstraction과 automation 이 직관적으로 잘 나타나있는 린덴마이어의 L-system 점화식, 진화알고리즘에 기반한 수학 함수의 최대최소 문제, 그리고 이를 응용한 주식 등 time series 분석과 플래피비드 등의 게임 분석이다. 이를 위하여 L-system 에는 티틀말을, 3차원 함수의 그래프에는 티틀크레프트를, 그리고 파이썬 코딩을 응합시켜 수학적 문제에 기반을 두는 코딩수학의 내용을 탐구한다

```
In [10]: from IPython.display import display
from IPython.display import Image

img = Image(filename='./ctsymbol.png', width='100%')
display(img)
```



```
In [ ]: 
```

```
In [ ]: 
```

((((z = f(x, y) == 최댓값과 최솟값))))

```
In [ ]: 
```

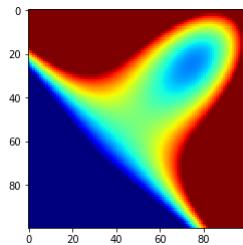
```
In [62]: import matplotlib.pyplot as plt
import numpy as np
%matplotlib inline

sz = 100
x,y = np.meshgrid(np.linspace(-2,2,sz),np.linspace(2,-2,sz))

z = x*x*x + y*y*y - 3*x*y

plt.imshow(z, vmin=-2, vmax=2, cmap='jet')
```

Out[62]: <matplotlib.image.AxesImage at 0x1051a70>



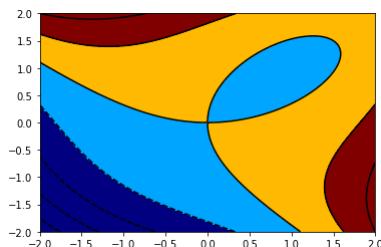
```
In [63]: import matplotlib.pyplot as plt
import numpy as np
%matplotlib inline

sz = 100
x,y = np.meshgrid(np.linspace(-2,2,sz),np.linspace(2,-2,sz))

z = x*x*x + y*y*y - 3*x*y

plt.contourf(x,y,z, vmin=-7, vmax=7, cmap='jet')
plt.contour(x,y,z,colors='black')
#plt.show()
```

Out[63]: <matplotlib.contour.QuadContourSet at 0x1157e6b0>



```
In [64]: import matplotlib.pyplot as plt
import numpy as np
%matplotlib inline

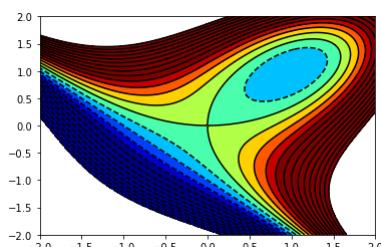
sz = 100
x,y = np.meshgrid(np.linspace(-2,2,sz),np.linspace(2,-2,sz))

z = x*x*x + y*y*y - 3*x*y

#plt.contourf(x,y,z, vmin=-7, vmax=7, cmap='jet')

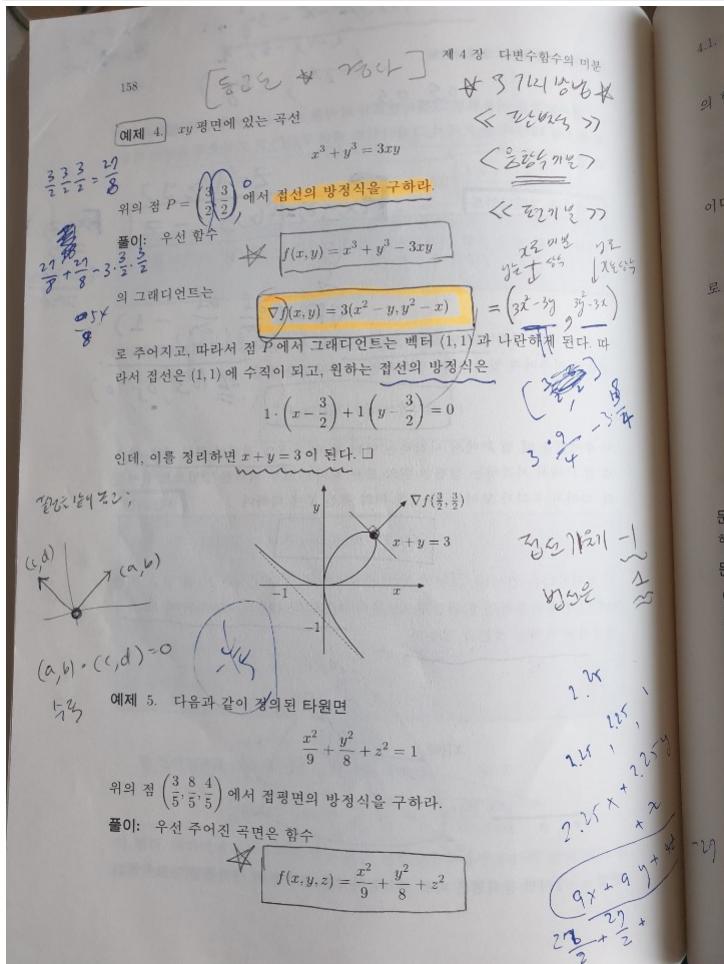
level = np.arange(-7,7,0.5)
plt.contourf(x,y,z, vmin=-2, vmax=2, cmap='jet', levels=level)
plt.contour(x,y,z,colors='black', levels=level)
#plt.show()
```

Out[64]: <matplotlib.contour.QuadContourSet at 0xf8ec690>



```
In [65]: from IPython.display import display
from IPython.display import Image

img = Image(filename='./alpha.jpg')
display( img )
```



```

In [66]: import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline

sz = 100
x,y = np.meshgrid(np.linspace(-2,2,sz),np.linspace(2,-2,sz))
# 난파이 메쉬그리드는 엑스축으로 먼저, 와이축으로 다음
# 행렬의 크기 순서와 meshgrid를 차이남

Z = x*x*x + y*y*y - 3*x*y

# plt.imshow( z, vmin=-1, vmax=1, cmap='jet' )

from IPython.display import clear_output
import time

np.random.seed(3)
nn = 7 # number of steps to take (and plot horizontally)
alpha = 0.03 # learning rate
sigma = 3 # standard deviation of the samples around current parameter vector

w = np.array([70.0, 30.0])
# start point : size=100 이므로 가로x 먼저 세로y 다음

#plt.ion() # something about plotting
#plt.figure(figsize=(10,7))
#plt.figure(figsize=(20,15))

prevx, prevy = [], []
for q in range(nn):
    plt.figure(figsize=(10,10))
    # draw the optimization landscape
    #ax1 = plt.subplot(1,nn,q+1)

    plt.imshow(Z, vmin=-1, vmax=1, cmap='jet')
    # draw a population of samples in black
    noise = np.random.randn(200, 2)
    wp = np.expand_dims(w, 0) + sigma*noise
    x,y = zip(*wp)

    plt.scatter(x,y,'k', edgecolors='face')

    # draw the current parameter vector in white
    plt.scatter([w[0]], [w[1]], 40, 'w', edgecolors='face')

    # draw estimated gradient as white arrow
    R = np.array( [ Z[ int(wi[1]), int(wi[0]) ] for wi in wp ] )
    R -= R.mean()
    R /= R.std() # standardize the rewards to be N(0,1) gaussian
    g = np.dot(R, noise)
    u = alpha * g
    plt.arrow( w[0], w[1], u[0], u[1], head_width=3, head_length=5, fc='w', ec='w')

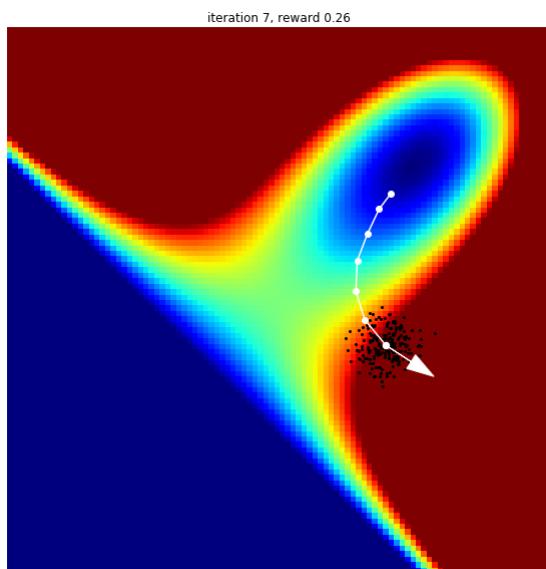
    plt.axis('off')
    plt.title("iteration %d, reward %.2f" % (q+1, Z[int(w[0]), int(w[1])]))

    # draw the history of optimization as a white line
    prevx.append(w[0])
    prevy.append(w[1])

    if len(prevx) > 0:
        plt.plot(prevx, prevy, 'wo-')

    w += u
    #plt.axis('tight')
    plt.pause(1.0)
    #time.sleep(1.0)
    clear_output(wait=True)

```

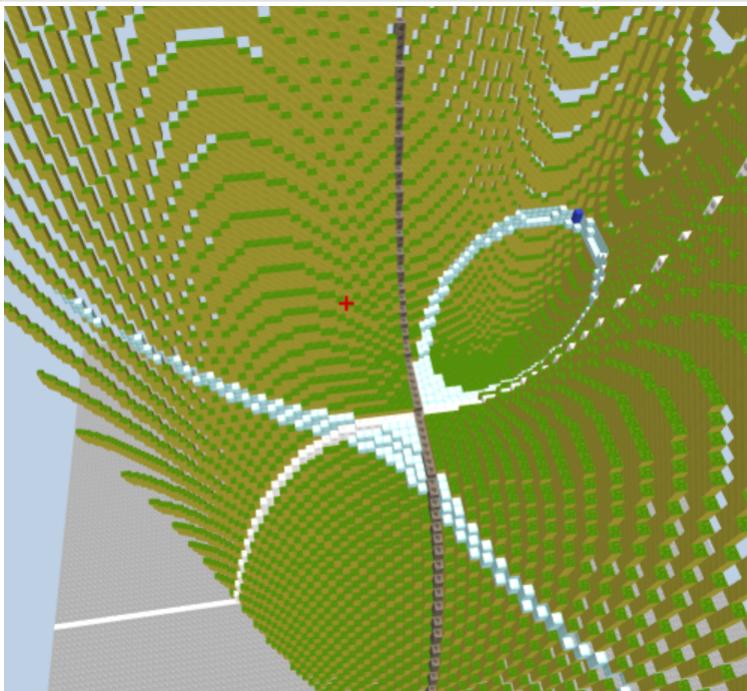


In []:

beginxyz 코딩

```
In [70]: from IPython.display import display
from IPython.display import Image

img = Image(filename='./alpha.png')
display( img )
```



```
In [18]: %%html
<pre>
beginxyz
x=0.05*x
y=0.05*y
z=0.05*z
// window(0.05) 로 대치 가능

function f(x,y) {
    return y* y* y + x* x* x - 3* x* y
}

if( Math.abs(f(x,y)-z) < 0.1 ) {
    if(x==0) item=20
    else if(y==0) item=10
    else if(x==1.5 && y==1.5) item=5
    else if(Math.abs(z)<0.1) item=6
    return item
}
return 0
</pre>

beginxyz
x=0.05*x
y=0.05*y
z=0.05*z
// window(0.05) 로 대치 가능

function f(x,y) {
    return y* y* y + x* x* x - 3* x* y
}

if( Math.abs(f(x,y)-z) < 0.1 ) {
    if(x==0) item=20
    else if(y==0) { item=10 }
    else if(x==1.5 && y==1.5) item=5
    else if(Math.abs(z)<0.1) item=6
    return item
}

return 0
```

```
In [19]: from IPython.display import *
HTML(
"""
<iframe src="http://www.javamath.com/snucode" width=1000 height=500> </iframe>
"""
)
# WinPython36F 터틀토리 안의 파이어폭스로 실행시키고,
# [begin] [실행] [터틀말] [터틀말 실행] 단추를 차례로 누르세요
```

Out[19]:

터틀말 + 터틀크래프트 :: 터북이 코딩수학
begin 실행 터틀말 터틀말 실행
<pre> 1 악습 말게임() { 2 mmmmmmm : m+++++m : 3 m_m_m_m : mooooom : 4 m__%_m : mm____mm : 5 mmmmmmm : 6 7 } 8 9 말게임 : 멀총 300 10 do uuddruuddl uuulr 11 do dddlu udrrrulduu 12 13 14 15 begincube 16 doit(sssulLssLuuss) 17 item=5 18 dovt([1,2,3,4,3,2,1]) 19 20 21 beginxyz 22 window(0.1) 23 if(xx*x + yy*y < 1 && z==5.5) return 6 24 </pre>

시흥-서울대 창의코딩 멘토링 ++ 수리정보 영재교육

== Part 0 =====

== 터틀크래프트 ++ 파이썬 코딩 ++ 다운받고 탐구 ==

A. [클릭 : 다운받기] 파이어폭스 ++ WinPython36F

B. [클릭 : 영재교육] 터틀크래프트 ++ 파이썬 코딩수학

□ <http://www.javamath.com/snucode/pythonxyz.pdf>

[클릭 : 읽으세요] 터틀말 + 터틀크래프트 설명서 및 교재

In []:

In [20]:

```
%html
<pre>

beginxyz

x=0.02*x
y=0.02*y
z=0.02*z
// window(0.02) 로 대체 가능

function f(x,y) {
return x* x* x - x* y* y
}

if( Math.abs(f(x,y)-z) < 0.02 ) {
  if(x==0) return 20
  return 10*x
}

return 0
</pre>

beginxyz

x=0.02*x
y=0.02*y
z=0.02*z
// window(0.02) 로 대체 가능

function f(x,y) {
return x* x* x - x* y* y
}

if( Math.abs(f(x,y)-z) < 0.02 ) {
  if(x==0) return 20
  return 10*y
}

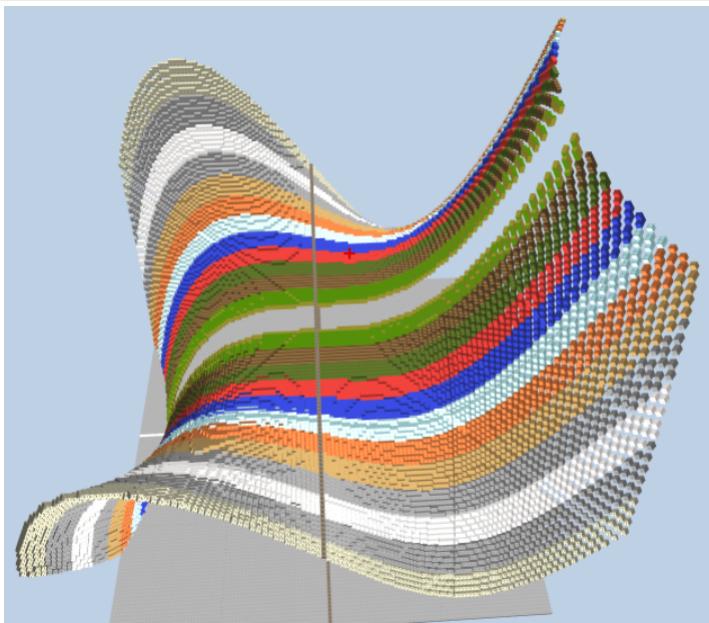
return 0
```

스토리코딩 탐구문제

스토리코딩 : 다음 $x(x+y)(x-y)$ 그래프의 MRI 사진을 설명하여라 !!

```
In [5]: from IPython.display import display
from IPython.display import Image

img = Image(filename='./x3-y2x.png')
display( img )
```



```
In [22]: %%html
```

```
<pre>
beginxyz

window(0.05)
// 이 영령의 뜻은 ??

function f(x,y) {
return (1-x)*(1-x)*Math.exp(-x*x-(y+1)*(y+1)) -
4*(x-x*x*x-y*y*y)*Math.exp(-x*x-y*y) - 2
}
```

```
if( Math.abs(f(x,y)-z) < 0.2 ) {
if(z==0 || z==1 || z==2) return 6
return 7
}

return 0
</pre>
```

```
beginxyz

window(0.05)
// 이 영령의 뜻은 ??

function f(x,y) {
```

```
return (1-x)*(1-x)*Math.exp(-x*x-(y+1)*(y+1)) -
4*(x-x*x*x-y*y*y)*Math.exp(-x*x-y*y) - 2
}
```

```
if( Math.abs(f(x,y)-z) < 0.2 ) {
if(z==0 || z==1 || z==2) return 6
return 7
}
```

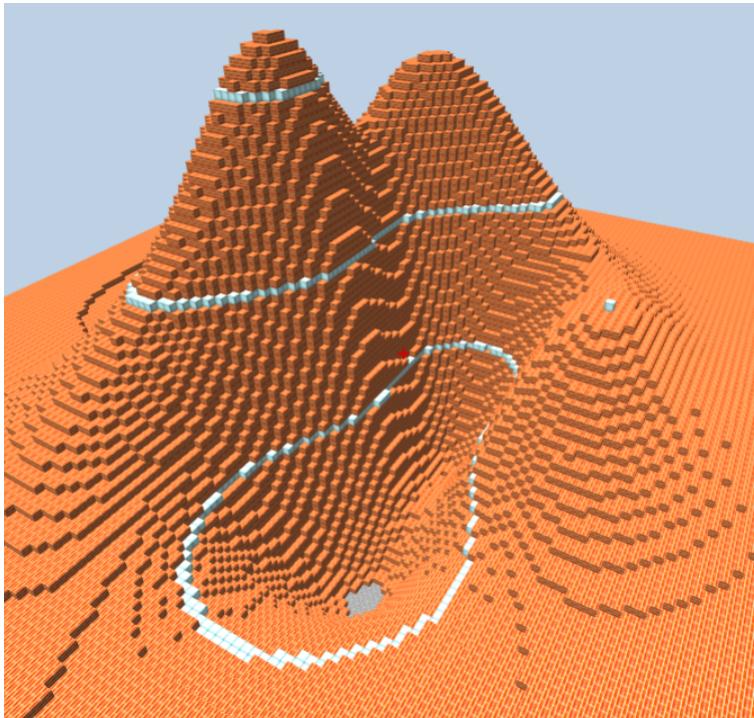
```
return 0
```

참고논문

<http://www.javamath.com/snuicode/handson.pdf> (<http://www.javamath.com/snuicode/handson.pdf>)

```
In [4]: from IPython.display import display
from IPython.display import Image

img = Image(filename='./3hills.png')
display( img )
```



저 위의 방법을 따라 python 으로 코딩하자 !!!

저 위의 방법으로 최댓값과 최솟값을 구해보자

```
In [23]: %%%html
<pre>
beginxyz

x=x*0.08
y=y*0.08
z=z*0.32

x=x+5
y=y+5
z=z+5
// 이 명령의 뜻은 ?

function f(x,y) {
return x*Math.sin(4*x) +
1.1*y*Math.sin(2*y)
}

if( Math.abs(f(x,y)-z) < 1.5 ) {
if( -13.7 <= z <= -13.5 ) return 6
if( z <= 6 && z >= 7 ) return 6
return 7
}

return 0

</pre>

beginxyz

x=x*0.08
y=y*0.08
z=z*0.32

x=x+5
y=y+5
z=z+5
// 이 명령의 뜻은 ?

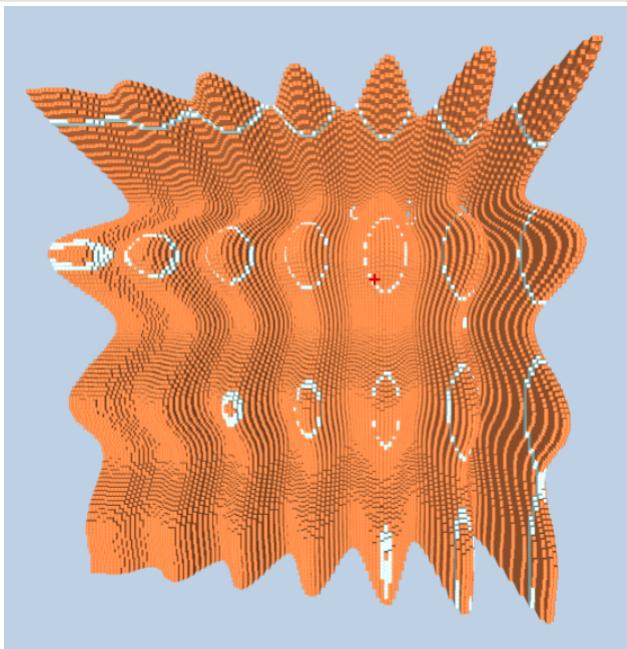
function f(x,y) {
return x*Math.sin(4*x) +
1.1*y*Math.sin(2*y)
}

if( Math.abs(f(x,y)-z) < 1.5 ) {
if( -13.7
```

참고논문

<http://www.javamath.com/snucode/lecture.pdf> (<http://www.javamath.com/snucode/lecture.pdf>)

```
In [6]: from IPython.display import display  
from IPython.display import Image  
  
img = Image(filename='./example.png')  
display( img )
```



참고논문

<http://www.javamath.com/snucode/intro.pdf> (<http://www.javamath.com/snucode/intro.pdf>)

In []:

아래의 방법을 따라 python 으로 코딩하자 !!!

아래의 방법으로 최댓값과 최솟값을 구해보자

In []:

In []:

In []:

((((y = f (x) == 최댓값과 최솟값)))))

In []:

```
In [27]: """
Visualize Genetic Algorithm to find a maximum point in a function.
"""


```

```
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline

DNA_SIZE = 10          # DNA length
POP_SIZE = 100         # population size
CROSS_RATE = 0.8       # mating probability (DNA crossover)
MUTATION_RATE = 0.003  # mutation probability
N_GENERATIONS = 50    # number of generations
X_BOUND = [0, 5]       # x upper and lower bounds

def F(x): return np.sin(10*x)*x + np.cos(2*x)*x      # to find the maximum of this function

# find non-zero fitness for selection
def get_fitness(pred): return pred + 1e-3 - np.min(pred)

# convert binary DNA to decimal and normalize it to a range(0, 5)
def translateDNA(pop): return pop.dot(2 ** np.arange(DNA_SIZE)[::-1]) / float(2**DNA_SIZE-1) * X_BOUND[1]

def select(pop, fitness):      # nature selection wrt pop's fitness
    idx = np.random.choice(np.arange(POP_SIZE), size=POP_SIZE, replace=True,
                           p=fitness/fitness.sum())
    return pop[idx]

def crossover(parent, pop):    # mating process (genes crossover)
    if np.random.rand() < CROSS_RATE:
        i_ = np.random.randint(0, POP_SIZE, size=1)
        cross_points = np.random.randint(0, 2, size=DNA_SIZE).astype(np.bool)
        parent[cross_points] = pop[i_, cross_points]
    return parent

def mutate(child):
    for point in range(DNA_SIZE):
        if np.random.rand() < MUTATION_RATE:
            child[point] = 1 if child[point] == 0 else 0
    return child

pop = np.random.randint(2, size=(POP_SIZE, DNA_SIZE))  # initialize the pop DNA

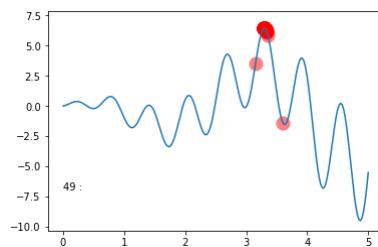
from IPython.display import clear_output
# plt.ion()           # something about plotting
x = np.linspace(*X_BOUND, 200)

for K in range(N_GENERATIONS):
    clear_output(wait=True)
    plt.plot(x, F(x))
    F_values = F(translateDNA(pop))  # compute function value by extracting DNA

    # something about plotting
    # if 'sca' in globals(): sca.remove()
    plt.scatter(translateDNA(pop), F_values, s=200, lw=0, c='red', alpha=0.5)

    # GA part (evolution)
    fitness = get_fitness(F_values)
    #print('Most fitted DNA: ', pop[np.argmax(fitness), :])
    pop = select(pop, fitness)
    pop_copy = pop.copy()
    for parent in pop:
        child = crossover(parent, pop_copy)
        child = mutate(child)
        parent[:] = child      # parent is replaced by its child
    plt.text(0, -7, '%d : %d' % (K, np.max(F_values)))
    plt.pause(0.05)

print("finished")
#plt.ioff(); plt.show()
```



```
finished
```

```
In [19]: """
```

```
(1/5)-ES with 1/5th success rule with visualization.
```

```
"""
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline

DNA_SIZE = 1           # DNA (real number)
DNA_BOUND = [0, 5]      # solution upper and lower bounds
N_GENERATIONS = 50
MUT_STRENGTH = 5.       # initial step size (dynamic mutation strength)

def F(x): return np.sin(10*x)*x + np.cos(2*x)*x    # to find the maximum of this function

# find non-zero fitness for selection
def get_fitness(pred): return pred.flatten()

def make_kid(parent):
    # no crossover, only mutation
    k = parent + MUT_STRENGTH * np.random.randn(DNA_SIZE)
    k = np.clip(k, *DNA_BOUND)
    return k

def kill_bad(parent, kid):
    global MUT_STRENGTH
    fp = get_fitness(F(parent))[0]
    fk = get_fitness(F(kid))[0]
    p_target = 1/5
    if fp < fk:    # kid better than parent
        parent = kid
        ps = 1.      # kid win -> ps = 1 (successful offspring)
    else:
        ps = 0.
    # adjust global mutation strength
    MUT_STRENGTH *= np.exp(1/np.sqrt(DNA_SIZE+1) * (ps - p_target)/(1 - p_target))
    return parent

parent = 5 * np.random.rand(DNA_SIZE)    # parent DNA

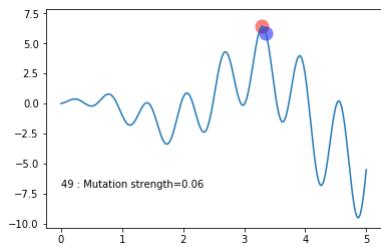
from IPython.display import clear_output
# import time
# plt.ion()      # something about plotting

x = np.linspace(*DNA_BOUND, 200)

for K in range(N_GENERATIONS):
    clear_output(wait=True)
    #plt.title('iteration %d' % K)
    # ES part
    kid = make_kid(parent)
    py, ky = F(parent), F(kid)      # for later plot
    parent = kill_bad(parent, kid)

    # something about plotting
    #plt.cla()
    plt.plot(x, F(x));
    plt.scatter(parent, py, s=200, lw=0, c='red', alpha=0.5)
    plt.scatter(kid, ky, s=200, lw=0, c='blue', alpha=0.5)
    plt.text(0, -7, '%d : Mutation strength=%2f' % (K, MUT_STRENGTH) )
    plt.pause(0.05)

print("finished")
# plt.ioff()
# plt.show()
```



```

In [3]: """
The Evolution Strategy can be summarized as the following term:
{mu/rho +, lambda}-ES
Here we use following term to find a maximum point.
{n_pop/n_pop + n_kid}-ES
"""

import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline

DNA_SIZE = 1           # DNA (real number)
DNA_BOUND = [0, 5]      # solution upper and lower bounds
N_GENERATIONS = 50
POP_SIZE = 100          # population size
N_KID = 50              # n kids per generation

def F(x): return np.sin(10*x)*x + np.cos(2*x)*x      # to find the maximum of this function

# find non-zero fitness for selection
def get_fitness(pred): return pred.flatten()

def make_kid(pop, n_kid):
    # generate empty kid holder
    kids = {'DNA': np.empty((n_kid, DNA_SIZE))}
    kids['mut_strength'] = np.empty_like(kids['DNA'])
    for kv, ks in zip(kids['DNA'], kids['mut_strength']):
        for kv1, ks1 in zip(kv, ks):
            # crossover (roughly half p1 and half p2)
            p1, p2 = np.random.choice(np.arange(POP_SIZE), size=2, replace=False)
            cp = np.random.randint(0, 2, DNA_SIZE, dtype=np.bool)  # crossover points
            kv1[cp] = pop['DNA'][p1, cp]
            kv1[~cp] = pop['DNA'][p2, ~cp]
            ks1[cp] = pop['mut_strength'][p1, cp]
            ks1[~cp] = pop['mut_strength'][p2, ~cp]

    # mutate (change DNA based on normal distribution)
    ks[:] = np.maximum(ks + (np.random.rand(*ks.shape)-0.5), 0.)  # must > 0
    kv := ks * np.random.randn(*kv.shape)
    kv[:] = np.clip(kv, *DNA_BOUND)  # clip the mutated value
    return kids

def kill_bad(pop, kids):
    # put pop and kids together
    for key in ['DNA', 'mut_strength']:
        pop[key] = np.vstack((pop[key], kids[key]))

    fitness = get_fitness(F(pop['DNA']))  # calculate global fitness
    idx = np.arange(pop['DNA'].shape[0])
    good_idx = idx[fitness.argsort()[-POP_SIZE:]]  # selected by fitness ranking (not value)
    for key in ['DNA', 'mut_strength']:
        pop[key] = pop[key][good_idx]
    return pop

pop = dict(DNA=5 * np.random.rand(1, DNA_SIZE).repeat(POP_SIZE, axis=0),  # initialize the pop DNA values
           mut_strength=np.random.rand(POP_SIZE, DNA_SIZE))  # initialize the pop mutation strength values

from IPython.display import clear_output
#plt.ion()  # something about plotting

x = np.linspace(*DNA_BOUND, 200)

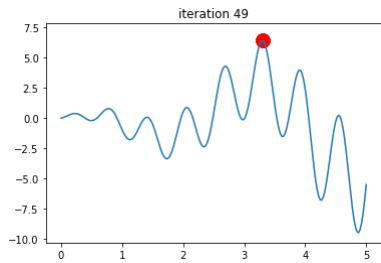
for K in range(N_GENERATIONS):
    clear_output(wait=True)
    plt.title('iteration %d' % K)

    # something about plotting
    # if 'sca' in globals(): sca.remove()
    plt.plot(x, F(x))
    sca = plt.scatter(pop['DNA'], F(pop['DNA']), s=200, lw=0, c='red', alpha=0.5); plt.pause(0.05)

    # ES part
    kids = make_kid(pop, N_KID)
    pop = kill_bad(pop, kids)  # keep some good parent for elitism
    #clear_output(wait=True)
    #plt.text(0, -7, '%d' % K)
    plt.pause(0.05)

print("finished")
# plt.ioff()
# plt.show()

```



finished

In []:

우리나라에서 초등학교에서 베이직 언어로 프로그래밍 의무교육이 시작된 것은 놀랍게도 1989년이다. 이때에도 전세계적으로 프로그래밍 교육이 선풍적으로 시작되었다. 그런데, 1997년부터 초등학교 3학년부터 영어교육 의무화가 실행되면서 프로그래밍 교육은 중단되었다. 그러다가 또다시 전세계적으로 코딩교육이 선풍적으로 일어나자 교육부는 허동지등 SW 소프트웨어 교육이라는 이상한 이름의 코딩교육을 2018년부터 실시한다는 말하고 있다. 외국에서는 1980년대부터 프로그래밍 교육을 계속하였고 이 영향으로 우리나라가 IMF를 맞았던 1998년도에 구글이라는 회사가 설립되어 세계를 지배하고 있다. 그렇다면 2018년 코딩교육과 1989년 프로그래밍 교육의 차이는 무엇일까? 저는 인공지능이 프로그래밍을 스스로하여 프로그램을 만드는 시대이다. 코딩은 기계가 하기 어려운 창의적인 컴퓨팅 사고력 (computational thinking) 역량을 키우는 것이다. 컴퓨팅 사고력이란 용어는 거북명령 프로그램 LOGO를 만들고 (그 후, 제자들이 원도우 환경에서 마우스 버전으로 만든 것이 고양이가 나오는 스크래치이다) 1989년에 XOR 문제 풀이로 인공지능 연구에 겨울이 오도록 한 MIT 대학의 수학자, 인공지능, 코딩교육자 패플릿 Papert의 마인드스톰 (1980년 출판) 책이다. 컴퓨팅 사고력의 핵심은 인간의 생각을 컴퓨터 기계가 이해하고 풀 수 있도록 문제를 표현하고 (Abstraction) 자동적으로 풀도록 (Automation) 코딩하는 것이다. 앞에서 소개한 진화 알고리즘은 문제를 유전자와 유전자 진화로 해결할 수 있도록 abstraction 표현하고 automation 자동으로 하는 서울대 수리정보 영재분야의 탐구는 abstraction과 automation이 직관적으로 잘 나타나있는 린덴마이어의 L-system 점화식, 진화알고리즘에 기반한 수학 핵수의 최대최소 문제, 그리고 이를 응용한 주식 등 time series 분석과 플래피버드 등의 개입 분석이다. 이를 위하여 L-system에는 티틀말을, 3차원 함수의 그레프에는 티틀크래프트를, 그리고 파이썬 코딩을 응합시켜 수학적 문제에 기반을 둔 코딩수학의 내용을 탐구한다. 다음은 인공지능의 역사와 코딩교육 언어에 대한 표이다. 인공지능 역사에서 1969년 인공지능의 겨울을 만든 패플릿은 그후 수학과 코딩교육을 위해 거북명령 프로그램 로고 (LOGO)를 만들고, 후에 마우스 버전인 스크래치 (scratch)로 발전하게 된다. 우리는 원래의 LOGO 버전으로 만들어진 티틀말과 마인크래프트와 같은 3차원 환경으로 코딩하는 티틀크래프트 그리고 파이썬 언어를 사용한다.

In [9]:

```
from IPython.display import display
from IPython.display import Image

img = Image(filename='./ctcoding.png', width='100%')
display( img )
```



In []:

In []:

==1969년에 인공지능의 겨울이 오고 인간은 달에 착륙하였다==

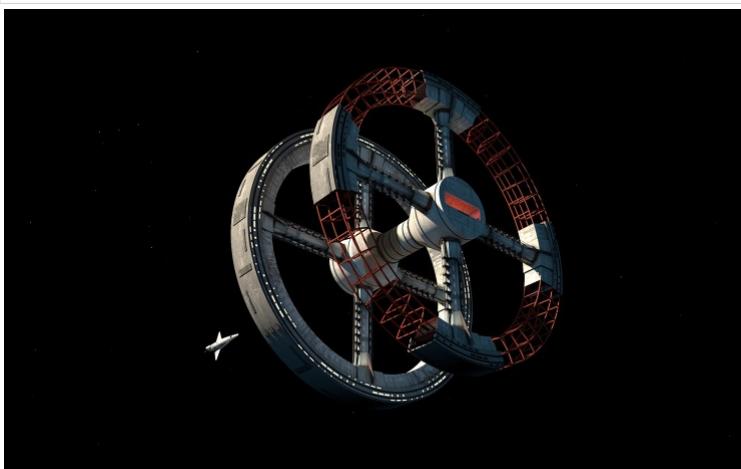
1968년에 나온 오딧세이 2001 영화에 인공지능 로봇과 우주선이

나온다. 창의코딩으로 다음 모양의 오딧세이 우주선을 만들어보자

In [7]:

```
from IPython.display import display
from IPython.display import Image

img = Image(filename='./torus.jpg')
display( img )
```



In []:

우주선 만들기 탐구과제

아래 코드의 x, y, z 좌표축을 변화시키며 우주선을 만들자

먼저 원통 통로를 만들고, 토러스 모양 붙혀서 만든다.

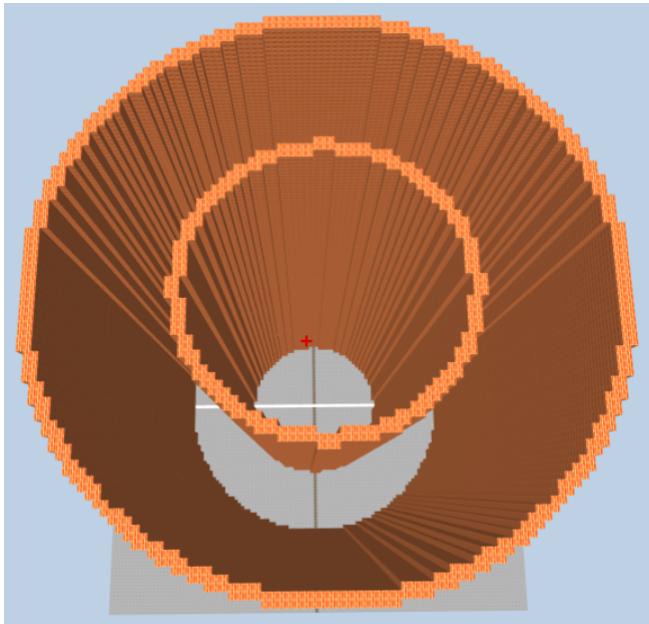
스토리코딩 탐구문제 :

우주선 통로에 해당하는 원통 모양을 만드는 스토리텔링 !!

```
In [1]: %%html
<pre>
beginxyz
R=30
r=10
s=r-2
D = Math.sqrt(x*x+y*y)
// 피타고라스 정리 사용
if( (D-R)*(D-R) < s*s ) return 0
if( (D-R)*(D-R) < r*r ) return 7
return 0
</pre>

beginxyz
R=30
r=10
s=r-2
D = Math.sqrt(x*x+y*y)
// 피타고라스 정리 사용
if( (D-R)*(D-R) < s*s ) return 0
if( (D-R)*(D-R) < r*r ) return 7
return 0
```

```
In [2]: from IPython.display import display
from IPython.display import Image
img = Image(filename='./twowon.png')
display( img )
```



스토리코딩 탐구문제

토러스 모양을 만드는 좌표식 코드를 스토리텔링 !!

```
In [3]: %%%html
<pre>
beginxyz
R=30
r=10
s=r-2

D = Math.sqrt(x*x+y*y)
// 피타고라스 정리 사용

if( (D-R)*(D-R) + z*z < s*s ) return 0
if( (D-R)*(D-R) + z*z < r*r ) return 7
return 0
</pre>
```

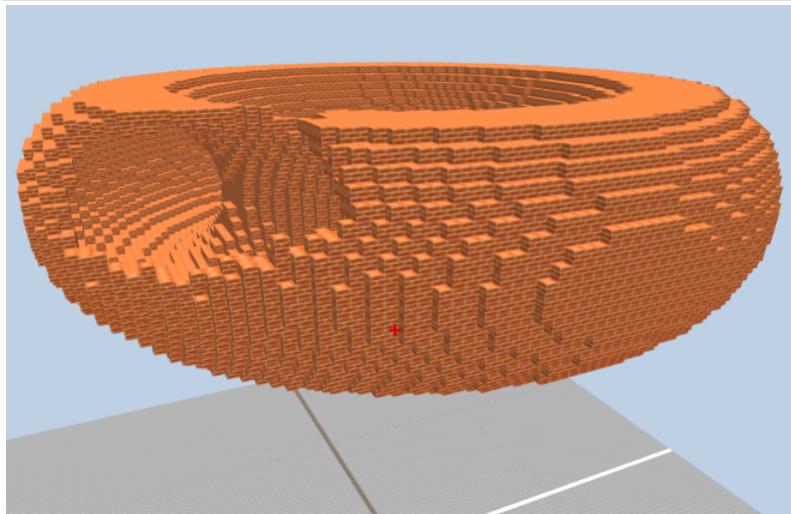
```
beginxyz
R=30
r=10
s=r-2

D = Math.sqrt(x*x+y*y)
// 피타고라스 정리 사용

if( (D-R)*(D-R) + z*z < s*s ) return 0
if( (D-R)*(D-R) + z*z < r*r ) return 7
return 0
```

```
In [48]: from IPython.display import display
from IPython.display import Image

img = Image(filename='./torusminus.png')
display( img )
```



스토리코딩 탐구문제

우주선 중앙의 비어있는 구 모양을 스토리텔링 !!

```
In [6]: %%%html
<pre>
beginxyz
R=30
r=10
s=1

d = Math.sqrt(x*x+y*y)
D = Math.sqrt(x*x+y*y+z*z)

if( D < R ) {
    if( d*d < r*r ) return 0
    if( (D-R)*(D-R) < s*s ) return 7
} else if( D < R+5 ) {
    if( (d-r)*(d-r) < s*s ) return 6
}

return 0
</pre>
```

```
beginxyz
R=30
r=10
s=1

d = Math.sqrt(x*x+y*y)
D = Math.sqrt(x*x+y*y+z*z)

if( D < R ) {

    if( d*d < r*r ) return 0
    if( (D-R)*(D-R) < s*s ) return 7

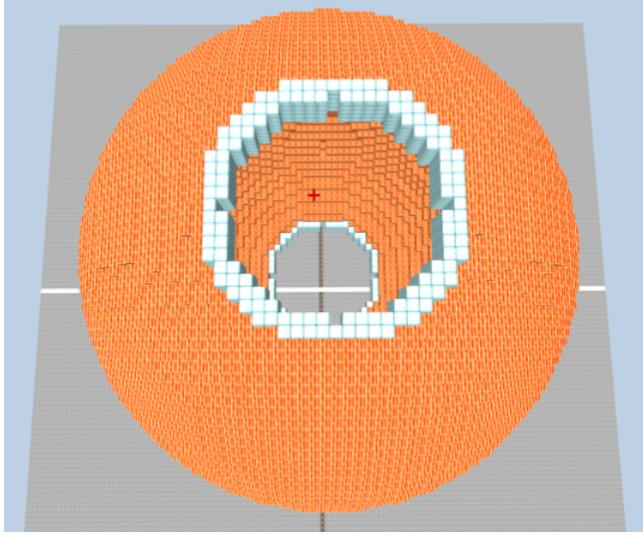
} else if( D < R+5 ) {

    if( (d-r)*(d-r) < s*s ) return 6
}

return 0
```

```
In [7]: from IPython.display import display
from IPython.display import Image

img = Image(filename='./station.png')
display( img )
```



In []:

In []:

참고 : 미로게임 스토리코딩

스토리코딩 : 다음 미로게임에 대한 코드를 설명하여라 !!

```

In [1]: """
    Visualize Genetic Algorithm to find a path to the target.
"""

import matplotlib.pyplot as plt
import numpy as np


from IPython.display import clear_output
%matplotlib inline


N_MOVES = 150
DNA_SIZE = N_MOVES*2           # 40 x moves, 40 y moves
DIRECTION_BOUND = [0, 1]
CROSS_RATE = 0.8
MUTATE_RATE = 0.0001
POP_SIZE = 100
N_GENERATIONS = 100
GOAL_POINT = [10, 5]
START_POINT = [0, 5]
OBSTACLE_LINE = np.array([[5, 2], [5, 8]])


class GA(object):
    def __init__(self, DNA_size, DNA_bound, cross_rate, mutation_rate, pop_size, ):
        self.DNA_size = DNA_size
        DNA_bound[1] += 1
        self.DNA_bound = DNA_bound
        self.cross_rate = cross_rate
        self.mutate_rate = mutation_rate
        self.pop_size = pop_size

        self.pop = np.random.randint(*DNA_bound, size=(pop_size, DNA_size))

    def DNA2product(self, DNA, n_moves, start_point):          # convert to readable string
        pop = (DNA - 0.5) / 2
        pop[:, 0], pop[:, n_moves] = start_point[0], start_point[1]
        lines_x = np.cumsum(pop[:, :n_moves], axis=1)
        lines_y = np.cumsum(pop[:, n_moves:], axis=1)
        return lines_x, lines_y

    def get_fitness(self, lines_x, lines_y, goal_point, obstacle_line):
        dist2goal = np.sqrt((goal_point[0] - lines_x[:, -1]) ** 2 + (goal_point[1] - lines_y[:, -1]) ** 2)
        fitness = np.power(1 / (dist2goal + 1), 2)
        points = (lines_x > obstacle_line[0, 0] - 0.5) & (lines_x < obstacle_line[1, 0] + 0.5)
        y_values = np.where(points, lines_y, np.zeros_like(lines_y) - 100)
        bad_lines = ((y_values > obstacle_line[0, 1]) & (y_values < obstacle_line[1, 1])).max(axis=1)
        fitness[bad_lines] = 1e-6
        return fitness

    def select(self, fitness):
        idx = np.random.choice(np.arange(self.pop_size), size=self.pop_size, replace=True, p=fitness/fitness.sum())
        return self.pop[idx]

    def crossover(self, parent, pop):
        if np.random.rand() < self.cross_rate:
            i_ = np.random.randint(0, self.pop_size, size=1) # select another individual from pop
            cross_points = np.random.randint(0, 2, self.DNA_size).astype(np.bool) # choose crossover points
            parent[cross_points] = pop[i_, cross_points] # mating and produce one child
        return parent

    def mutate(self, child):
        for point in range(self.DNA_size):
            if np.random.rand() < self.mutate_rate:
                child[point] = np.random.randint(*self.DNA_bound)
        return child

    def evolve(self, fitness):
        pop = self.select(fitness)
        pop_copy = pop.copy()
        for parent in pop: # for every parent
            child = self.crossover(parent, pop_copy)
            child = self.mutate(child)
            parent[:] = child
        self.pop = pop


class Line(object):
    def __init__(self, n_moves, goal_point, start_point, obstacle_line):
        self.n_moves = n_moves
        self.goal_point = goal_point
        self.start_point = start_point
        self.obstacle_line = obstacle_line

        plt.ion()

    def plotting(self, lines_x, lines_y):
        plt.cla()
        plt.scatter(*self.goal_point, s=200, c='r')
        plt.scatter(*self.start_point, s=100, c='b')
        plt.plot(self.obstacle_line[:, 0], self.obstacle_line[:, 1], lw=3, c='k')
        plt.plot(lines_x.T, lines_y.T, c='k')
        plt.xlim(-5, 15)
        plt.ylim(-5, 15)
        plt.pause(0.01)

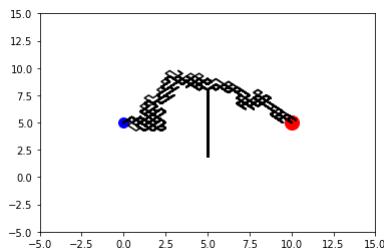
ga = GA(DNA_size=DNA_SIZE, DNA_bound=DIRECTION_BOUND,
         cross_rate=CROSS_RATE, mutation_rate=MUTATE_RATE, pop_size=POP_SIZE)

env = Line(N_MOVES, GOAL_POINT, START_POINT, OBSTACLE_LINE)

for generation in range(N_GENERATIONS):
    clear_output(wait=True)
    ix, iy = ga.DNA2product(ga.pop, N_MOVES, START_POINT)
    fitness = ga.get_fitness(ix, iy, GOAL_POINT, OBSTACLE_LINE)
    ga.evolve(fitness)
    #print('Gen:', generation, '| best fit:', fitness.max())
    env.plotting(ix, iy)
    plt.pause(0.05)

#plt.ioff()
#plt.show()

```



In []:

참고 : 다음 코드에는 tensorflow 설치가 필요

WinPython36F에서는 tensorflow 설치가 안됩니다 !!!

```

In [14]:
"""
The basic idea about Nature Evolution Strategy with visualization.

Dependencies:
Tensorflow >= r1.2
numpy
matplotlib
"""

import numpy as np
import matplotlib.pyplot as plt
import tensorflow as tf
from tensorflow.contrib.distributions import MultivariateNormalFullCovariance
%matplotlib inline

DNA_SIZE = 2          # parameter (solution) number
N_POP = 20            # population size
N_GENERATION = 50    # training step
LR = 0.02             # learning rate

# fitness function
def get_fitness(pred): return -((pred[:, 0])**2 + pred[:, 1]**2)

# build multivariate distribution
mean = tf.Variable(tf.random_normal([2, 1], 13., 1.), dtype=tf.float32)
cov = tf.Variable(5., * tf.eye(DNA_SIZE), dtype=tf.float32)
mvn = MultivariateNormalFullCovariance(loc=mean, covariance_matrix=cov)
make_kid = mvn.sample(N_POP)           # sampling operation

# compute gradient and update mean and covariance matrix from sample and fitness
tfkids_fit = tf.placeholder(tf.float32, [N_POP, 1])
tfkids = tf.placeholder(tf.float32, [N_POP, DNA_SIZE])
loss = -tf.reduce_mean(mvn.log_prob(tfkids)*tfkids_fit)      # log prob * fitness
train_op = tf.train.GradientDescentOptimizer(LR).minimize(loss) # compute and apply gradients for mean and cov

sess = tf.Session()
sess.run(tf.global_variables_initializer())                      # initialize tf variables

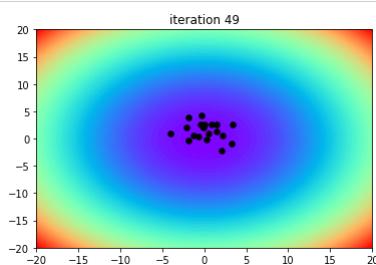
# something about plotting (can be ignored)
n = 300
x = np.linspace(-20, 20, n)
X, Y = np.meshgrid(x, x)
Z = np.zeros_like(X)
for i in range(n):
    for j in range(n):
        Z[i, j] = get_fitness(np.array([[x[i], x[j]]]))
# plt.contourf(X, Y, -Z, 100, cmap=plt.cm.rainbow); plt.ylim(-20, 20); plt.xlim(-20, 20); plt.ion()
from IPython.display import clear_output

# training
for g in range(N_GENERATION):
    clear_output(wait=True)
    plt.title('Iteration %d' % g)
    plt.contourf(X, Y, -Z, 100, cmap=plt.cm.rainbow);
    plt.ylim(-20, 20); plt.xlim(-20, 20)
    kids = sess.run(make_kid)
    kids_fit = get_fitness(kids)
    sess.run(train_op, {tfkids_fit: kids_fit, tfkids: kids})    # update distribution parameters

    # plotting update
    #if 'sca' in globals(): sca.remove()
    #sca = plt.scatter(kids[:, 0], kids[:, 1], s=30, c='k'); plt.pause(0.01)
    plt.scatter(kids[:, 0], kids[:, 1], s=30, c='k');
    plt.pause(0.01)

print('Finished');
# plt.ioff(); plt.show()

```



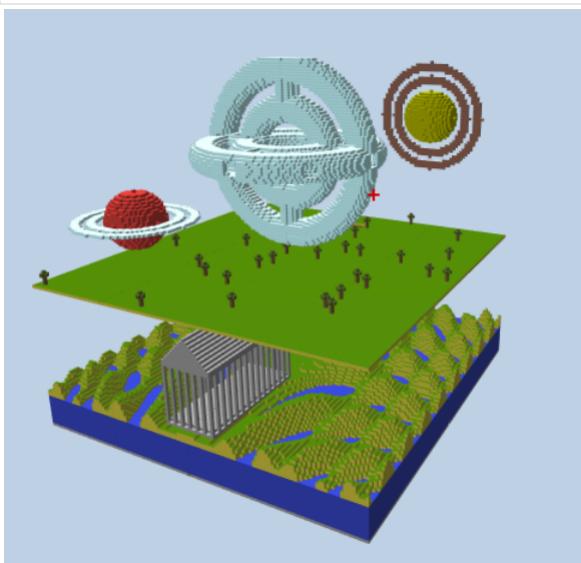
Finished

```
In [ ]:
```

수리정보 영재교육 탐구의 예 :

```
In [ ]:
```

```
In [24]: from IPython.display import display  
from IPython.display import Image  
  
img = Image(filename='./sasa.png')  
display( img )
```



3차원 $f(x,y)$ 탐구에 진화알고리즘 적용

3차원 $f(x,y)$ 탐구에 터틀크래프트 적용

2차원 $y=f(x)$ 경우의 진화알고리즘 분석

주식 등 시계열분석에 진화알고리즘 적용

플래피버드 등 진화알고리즘과 게임 탐구

```
In [ ]:
```

타임시리즈 (TimeSeries) 주가 그래프 분석

<http://www.javamath.com/snucode/구글투자-NES.htm> (<http://www.javamath.com/snucode/구글투자-NES.htm>)

뉴로이볼류션 (NeuroEvolution 플래피버드)

쿠키런 등의 게임을 플래피버드 코드로 만들어보자

```
In [13]: from IPython.display import *
HTML(
"""
<iframe src="http://www.javamath.com/snucode/flappybird.htm" width=500 height=500></iframe>
"""
)
```

Out[13]:

