

2023학년도

시소프트웨어학과 학술제 & 홈커밍데이

일시: 2023.11.24.(금) 09:30~16:30

장소: 원화관 205, 603, 604, 605

선문대학교 시소프트웨어학과

2023 시소프트웨어학과 학술제 & 홈커밍 데이

학회장 서준철



행사 주의사항!

- 모든 행사 참가 시 서명 필수(전공 수업출석 대체)
>> 교양수업은 출석 불가(사전 협의 필요)
- 사람이 많아 혼잡하니 최대한 질서정연하게!!
- 마지막에 추첨을 통한 경품지급 행사 있으니
중간에 이탈하지 않기..

- 학생회 경품 추첨시 학회비 미납부자는 경품 수령 불가

> 학과에서 준비한 소정의 상품으로 대체

- 골든벨, 알고리즘 경진대회, 프로젝트 발표
상금은 학회비 미납부자도 수령 가능

- 점심식사때 사람이 많이 몰리니
질서정연하게 싸인하고 수령하기

- 쓰레기 막 버리지 말고 꼭!!!!
정해진곳에 버리기



일정 안내

일정 안내

시간	일정	장소	지도교수	행사진행	행사보조
9 : 30 ~ 9 : 50	행사 개회식 일정소개	원화관 205	학과교수	학생회장	학생회
10 : 00 ~ 11 : 00	도전골든벨(1학년)	원화관 603	이영한	←	1학년 과대표
10 : 00 ~ 12 : 00	알고리즘 경진대회(2~4학년)	원화관 205	황세웅	←	2,3,4학년 과대표
12 : 00 ~ 12 : 30	점심식사	원화관 205, 603	-	김아름 선생님	학생회, 각 학년 과대표
12 : 30 ~ 14 : 00	초청강연 #1(이소영 이사)	원화관 205	이영한	←	
12 : 30 ~ 14 : 00	초청강연 #2(신현묵 이사)	원화관 603	황석형	←	
14 : 00 ~ 15 : 30	팀프로젝트 작품발표회(3학년)	원화관 604	황세웅, 김홍수	←	3,4학년 과대표
14 : 00 ~ 15 : 30	팀프로젝트 작품발표회(4학년)	원화관 605			
15 : 30 ~ 16 : 00	<u>HomeComingDay</u> 선후배 토크콘서트	원화관 205	-	학생회장	학생회
16 : 00 ~ 16 : 30	경품추첨 및 시상식(제비뽑기 및 행사 상품 수령)	원화관 205	학과교수	학생회장	학생회
16 : 30 ~	폐회식	원화관 205	학과교수	학생회장	학생회



일정 안내 - 1학년

시간

10 : 00 ~ 11 : 00

일정

도전 골든벨

장소

원화관 603호

일정 안내 - 2,3,4학년

시간

10 : 00 ~ 12 : 00

일정

알고리즘
경진대회

장소

원화관 205호

시간	일정	장소
11 : 30 ~ 12 : 30	점심 식사	원화관 205앞, 원화관 606B 앞

점심식사

1. 학술제 참가 서명
2. 경품 추첨 용지 배분
3. 마이크로소프트 이사님 특강 듣는 사람은 12시 20분에 205호
4. 신현묵 이사님 특강 듣는 사람은 11시40분부터 원화관 606B

일정 안내

시간	일정	장소
12 : 30 ~ 14 : 00	초청강연 #1	원화관 205
12 : 30 ~ 14 : 00	초청강연 #2	원화관 603

시간	일정	장소
12 : 30 ~ 14 : 00	초청강연 #1	원화관 205

마이크로 소프트 이소영 이사님 특강

IT산업트렌드와진로, SW프로젝트응용, 확률과통계,
컴퓨터보안, 웹인텔리전스, 산학협력SW프로젝트

일정 안내

시간	일정	장소
12 : 30 ~ 14 : 00	초청강연 #2	원화관 603

우리들 녹지 국제병원 CIO
신현묵 이사님 특강(의료 AI)

소프트웨어품질관리, 소프트웨어분석설계, 컴퓨터알고리즘,
인공지능공학_11반, 산학협력SW프로젝트

일정 안내 - 3,4학년

시간	일정	장소
14 : 00 ~ 15 : 30	팀프로젝트 작품 발표회 - 3학년	원화관 604
14 : 00 ~ 15 : 30	팀프로젝트 작품 발표회 - 4학년	원화관 605

일정 안내

시간	일정	장소
15 : 30 ~ 16 : 00	Home Coming Day 행사 진행	원화관 205

졸업생 선배 초청(직장인, 대학원생)

**졸업 후 생활, 학교생활&학점관리 팁 등
토크콘서트 형식으로 진행**

시간

일정

장소

16 : 00 ~ 16 : 30

경품 추첨 및 시상

원화관 205

경품 수령은 모든 일정이 끝난 후 606B에서 진행

일정 안내

시간

16 : 30 ~

일정

폐회식 및
사진촬영

장소

원화관 205

Q&A

```
def __init__(self, n_test, self.weights):
    in_train_data = len(training_data)
    in_test_data = len(test_data)
    random.shuffle(training_data)
    mini_batches = [
        training_data[k:k+mini_batch_size]
        for k in xrange(0, in_train_data, mini_batch_size)
    ]
    for mini_batch in mini_batches:
        self.update_mini_batch(mini_batch, eta)
    if test_data:
        print "Epoch %d: Training Accuracy: %f" % (j, self.evaluate(training_data, self.weights))
    else:
        print "Epoch %d: Training Accuracy: %f" % (j, self.evaluate(training_data, self.weights))
        print "Epoch %d: Test Accuracy: %f" % (j, self.evaluate(test_data, self.weights))

def update_mini_batch(self, mini_batch, eta):
    n_test = len(test_data)
    n_train = len(training_data)
    n_test_batches = n_test / mini_batch_size
    n_train_batches = n_train / mini_batch_size
    for b in xrange(n_test_batches):
        x, y = test_data[b*mini_batch_size:(b+1)*mini_batch_size]
        delta_nabla_b, delta_nabla_w = self.propagate_gradients(x, y)
        nabla_b = [nb+dnb for nb, dnb in zip(self.biases, delta_nabla_b)]
        nabla_w = [nw+dnw for nw, dnw in zip(self.weights, delta_nabla_w)]
        self.weights = [w-(eta*dnw) for w, dnw in zip(self.weights, nabla_w)]
        self.biases = [b-(eta*dnb) for b, dnb in zip(self.biases, nabla_b)]

def propagate_gradients(self, x, y):
    nabla_b = [np.zeros(b.shape) for b in self.biases]
    nabla_w = [np.zeros(w.shape) for w in self.weights]
    # feedforward
    activation = x
    activations = [x] # list to store all the activations, starting from bias input
    for b, w in zip(self.biases, self.weights):
        z = np.dot(w, activation)+b
        zs.append(z)
        activation = sigmoid(z)
        activations.append(activation)
    # backward pass
    delta = self.cost_derivative(activations[-1]-y) # delta at output node
    # delta at hidden nodes
    sigmoid_prime = self.sigmoid_prime(zs[-1])
    delta_b[-1] = delta
    delta_w[-1] = np.dot(delta, activations[-2].transpose())
    # range(2, self.num_layers)
    for l in range(2, self.num_layers):
        prime_z = sigmoid_prime(zs[l-1])
        delta = prime_z * self.weights[l-1].transpose()
        delta_b[l-1] = delta
        delta_w[l-1] = np.dot(delta, activations[l-2].transpose())
    return (nabla_b, nabla_w)
```

Multiple hidden Layers

Output Layer