



# Python Programming

## Лекц 11

Багш Ж.Золжаргал

Х. Хулан

# EXCEPTIONS and ASSERTIONS



- Гэнэтийн нөхцөл байдалд хүрэхэд юу болох вэ?
- Жагсаалтын хязгаараас хэтрэх

```
test = [1, 7, 4]
```

```
test[4]
```

→ `IndexError`

- Тохиромжгүй төрлийг хөрвүүлэхийг оролдох

```
int(test)
```

→ `TypeError`

- Байхгүй хувьсагч руу хандах

```
a
```

→ `NameError`

# EXCEPTIONS and ASSERTIONS



- Нийтлэг алдааны төрлүүд
  - `SyntaxError`: Паятон програм хөрвүүлж чадахгүй
  - `NameError`: Локал болон Глобал хувьсагч байхгүй
  - `AttributeError`: Шинж чанар байхгүй
  - `TypeError`: үйлдэл хийх зөв төрөл байхгүй
  - `ValueError`: Төрөл зөв боловч, Утгын алдаа
  - `IOError`: ОГаралт систем буруу ажиллаж байгааг мэдээлнэ. (файл олдохгүй гэх мэт)

# DEALING WITH EXCEPTIONS



- Паятон код нь онцгой тохиолдлыг зохицуулагчаар хангаж өгдөг.

```
try:
    a = int(input("Tell me one number:"))
    b = int(input("Tell me another number:"))
    print(a/b)
except:
    print("Bug in user input.")
```

# HANDLING SPECIFIC EXCEPTIONS



- Тодорхой төрлийн онцгой тохиолдлуудыг тусад нь заана

```
try:
    a = int(input("Tell me one number: "))
    b = int(input("Tell me another number: "))
    print("a/b = ", a/b)
    print("a+b = ", a+b)
except ValueError:
    print("Could not convert to a number.")
except ZeroDivisionError:
    print("Can't divide by zero")
except:
    print("Something went very wrong.")
```

Зөвхөн эдгээр  
алдаа илрэхэд

Бусад алдаанд

# OTHER EXCEPTIONS



- else:
  - Try блокын нэг ч онцгой тохиолдол үүсээгүй үед гүйцэтгэгдэнэ
- finally:
  - try, else, except блокуудын дараа дандаа гүйцэтгэгдэнэ.
  - Юу ч болсон хамаагүй ажиллах буюу цэвэрлэх кодонд хэрэглэдэг. (Ж нь: файл хаах)

# WHAT TO DO WITH EXCEPTIONS?



- Алдаа гарахад юу хийх хэрэгтэй вэ?
- Чимээгүй алдаа
  - Анхдагч утгыг олгох эсвэл үргэлжлүүлэх
  - Муу санаа! Хэрэглэгч анхааруулга авахгүй
- Алдаа болон утгыг буцаах
  - Ямар утга сонгох?
  - Онцгой утгыг шалгах цогц код
- Програмыг зогсоох, signal error нөхцөл
  - Паятонд: онцгой тохиолдол үүсгэх

```
raise Exception("descriptive string")
```

# EXCEPTIONS AS CONTROL FLOW



- Алдаа гарсан үед онцгой утгыг буцаахгүй, алдааны утгыг буцаана
- Оронд нь, Зөв үр дүн буцаах боломжгүй үед онцгой дохиолдол үүсгэх

```
raise <exceptionName>(<arguments>)
```

```
raise ValueError("something is wrong")
```


Түлхүүр үг

Үүсгэхийг хүссэн  
алдааны нэр

Заавал биш, гэвч  
ихэвчлэн мессэж авна



# EXAMPLE: RAISING AN EXCEPTION



```
def get_ratios(L1, L2):  
    """ Assumes: L1 and L2 are lists of equal length of numbers  
        Returns: a list containing L1[i]/L2[i] """  
    ratios = []  
    for index in range(len(L1)):  
        try:  
            ratios.append(L1[index]/L2[index])  
        except ZeroDivisionError:  
            ratios.append(float('nan')) #nan = not a number  
        except:  
            raise ValueError('get_ratios called with bad arg')  
    return ratios
```


Програмын урсгалыг  
өөрийн алдааг үүсгэж  
удирдах

# Lambda



- Паятон хэлэнд нэргүй функцыг ламбда (lambda) түлхүүр үгээр тодорхойлдог.
- Ламбда функц нь энгийн функцээс бага зэрэг ялгаатай бөгөөд дараах шинж чанартай:
  - олон параметр авах бөгөөд зөвхөн ганц л илэрхийлэл агуулах боломжтой
  - Ламбда функцыг функцын объектуудыг буцааж өгөхөд ашиглаж болно
  - Синтаксын хувьд зөвхөн ганц илэрхийллээр хязгаарлагддаг.

# Lambda



```
# энгийн функц
def adder(x, y):
    return x + y

# lambda функц
adder1 = lambda x, y: x + y

print("adder", adder(3, 4))
Print("adder1", adder1(3, 4))
```

Үр дүн

```
('adder', 7)
('adder1', 7)
```


# Filter



- Шүүлтүүр функц нь элементүүдийн цуглуулгаас элементүүдийг шүүхэд ашиглагдах бөгөөд шүүх функц авдаг
- Үр дүнд нь зөвхөн шүүх функцээр сонгосон элементүүдийг агуулсан шинэ давтагдах элементүүд байна
- Синтакс нь:

```
filter(function, iterable)
```

# Filter



```
Data = [1, 3, 5, 2, 7, 4, 10]
print('data:', data)
def is_even(i):
    return i % 2 == 0
# Функц ашиглан тэгш тоонуудыг шүүх
d2 = list(filter(is_even, data))
print('d2:', d2)
```

Үр дүн

```
Data: [1, 3, 5, 2, 7, 4, 10]
d2: [2, 4, 10]
```

# Filter



- Ламбда ашигласан жишээ

```
data = [1, 3, 5, 2, 7, 4, 10]
print('data:', data)
# Ламбда функц ашиглан тэгш тоонуудыг шүүх
d1 = list(filter(lambda i: i % 2 == 0, data))
print('d1:', d1)
```

# Filter

```
class Person:
    def __init__(self, name, age):
        self.name = name
        self.age = age
    def __str__(self):
        return 'Person(' + self.name + ', ' + str(self.age) +
            ')\n'

data = [Person('Alun', 54), Person('Niki', 21), Person('Megan',
19)]
for p in data:
    print(p, end=', ')
    print('\n-----')
# 21-ээс доош насны хүмүүсийг шүүхэд ламбда функц ашиглав
d3 = list(filter(lambda p: p.age <= 21, data))
for p in d3:
    print(p, end=', ')
```

# Filter



- Үрдүн

```
Person(Alun, 54), Person(Niki, 21), Person(Megan, 19),  
-----  
Person(Niki, 21), Person(Megan, 19),
```




# Map



- Map функц нь өгөгдсөн функцыг давтагдах бүх элементэд гүйцэтгэж өгдөг.
- Энэ нь ашигласан функцээс үүссэн үр дүнгийн шинэ давталтыг буцааж өгдөг.
- Map функцын синтакс

```
map(function, iterable, ...)
```

# Map



```
data = [1, 3, 5, 2, 7, 4, 10]
print('data:', data)
# Жагсаалтын элемент бүр дээр ламбда функц хэрэгжихдээ
# map функцээр дамжина
d1 = list(map(lambda i: i + 1, data))
print('d1', d1)
```

Үр дүн

```
data: [1, 3, 5, 2, 7, 4, 10]
d1 [2, 4, 6, 3, 8, 5, 11]
```

# Map



- Хоёр жагсаалтын тоонуудыг хооронд нь нэмэх

```
data1 = [1, 3, 5, 7]
data2 = [2, 4, 6, 8]
result = list(map(lambda x, y: x + y, data1, data2))
print(result)
```

Үр дүн

```
[3, 7, 11, 15]
```

# Reduce



- Reduce функц нь функцийг давтагдах элементэд ашиглах бөгөөд элемент тус бүрд буцаж ирсэн үр дүнг нэгтгэн нэг үр дүн болгон авдаг.
- Reduce функц нь паятон 2-ын цөмд агуулагдсан байдаг боловч паятон 3-ийн цөмд агуулагдаагүй.
- Reduce функцын хэрэглээ маш их байдаг ба паятон 3 дээр functools модулийг импорт (import) хийн оруулж ирсэнээр ашиглаж болно.

# Reduce



```
from functools import reduce
data = [1, 3, 5, 2, 7, 4, 10]
result = reduce(lambda total, value: total + value, data)
print(result)
```

- Үр дүн: 32

# Reduce



```
data = [Person('John', 54), Person('Phoebe', 21),  
        Person('Adam', 19)]  
total_age = reduce(lambda running_total, person: running_total  
+ person.age, data, 0)  
average_age = total_age // len(data)  
print('Average age:', average_age)
```

- Үр дүн: 31