Live Functional Programming with Typed Holes

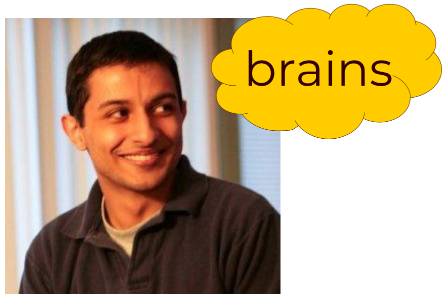
Cyrus Omar

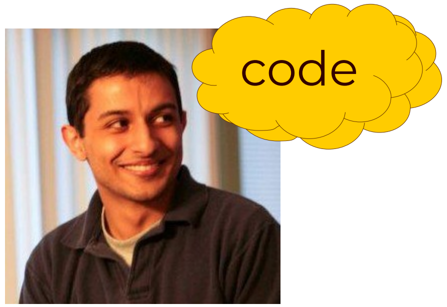
Future of Programming Lab (FP Lab)

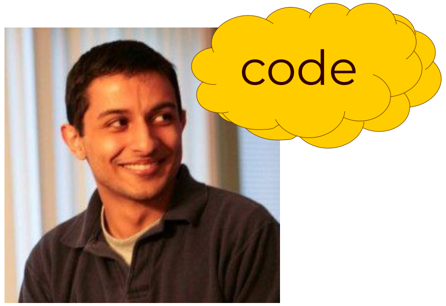
University of Michigan

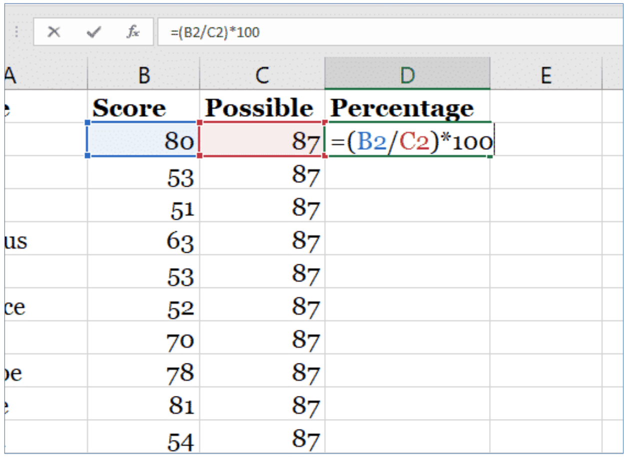
@neurocy

http://fplab.mplse.org/

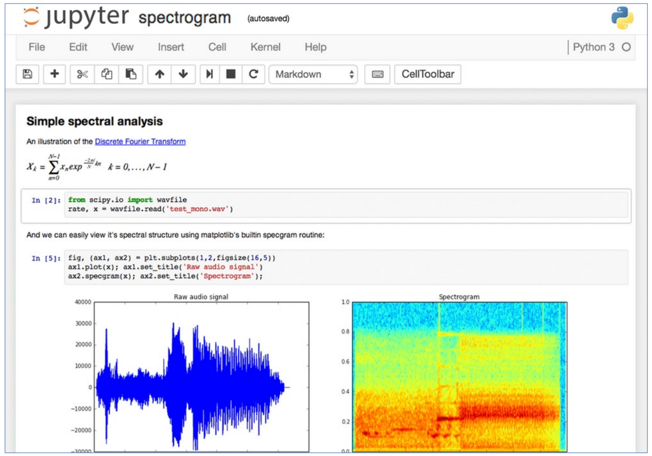

brains


code




=(B2/C2)*100

A	B	C	D	E
e	Score	Possible	Percentage	
	80	87	=(B2/C2)*100	
	53	87		
	51	87		
us	63	87		
	53	87		
ce	52	87		
	70	87		
pe	78	87		
e	81	87		
	54	87		


jupyter spectrogram (autosaved)
File Edit View Insert Cell Kernel Help
Python 3
Markdown
CellToolbar

Simple spectral analysis

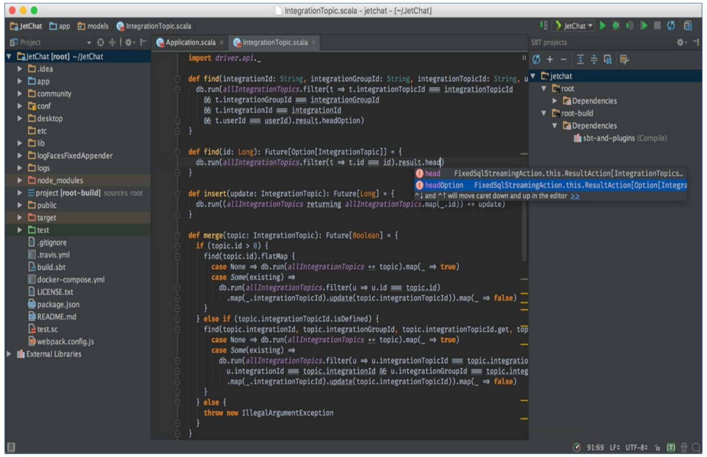
An illustration of the Discrete Fourier Transform

$$X_k = \sum_{n=0}^{N-1} x_n exp^{\frac{-2\pi i}{N}kn} \quad k = 0, \dots, N-1$$

```
In [2]: from scipy.io import wavfile
        rate, x = wavfile.read('test_mono.wav')
```

And we can easily view it's spectral structure using matplotlib's builtin specgram routine:

```
In [5]: fig, (ax1, ax2) = plt.subplots(1,2,figsize(16,5))
        ax1.plot(x); ax1.set_title('Raw audio signal')
        ax2.specgram(x); ax2.set_title('Spectrogram');
```

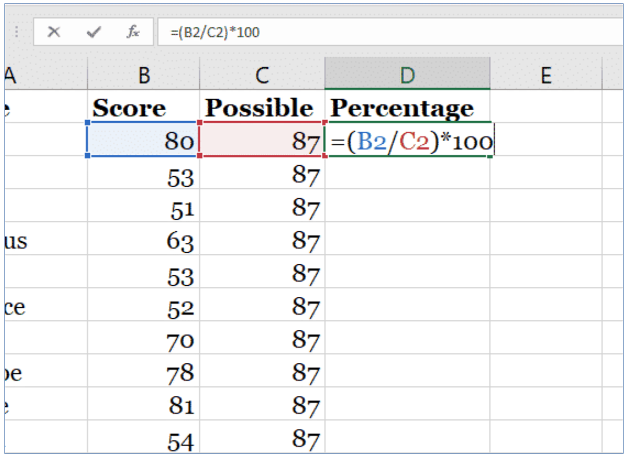


```
import driver.api._

def find(integrationId: String, integrationGroupId: String, integrationTopicId: String, u
  db.run(allIntegrationTopics.filter(t ⇒ t.integrationTopicId == integrationTopicId
    && t.integrationGroupId == integrationGroupId
    && t.integrationId == integrationId
    && t.userId == userId).result.headOption)
}

def find(id: Long): Future[Option[IntegrationTopic]] = {
  db.run(allIntegrationTopics.filter(t ⇒ t.id == id).result.head)
}                                  head        FixedSqlStreamingAction.this.ResultAction[IntegrationTopics…
                                   headOption  FixedSqlStreamingAction.this.ResultAction[Option[Integra
def insert(update: IntegrationTopic): Future[Long] = {   ^↓ and ^↑ will move caret down and up in the editor >>
  db.run((allIntegrationTopics returning allIntegrationTopics.map(_.id)) += update)
}

def merge(topic: IntegrationTopic): Future[Boolean] = {
  if (topic.id > 0) {
    find(topic.id).flatMap {
      case None ⇒ db.run(allIntegrationTopics += topic).map(_ ⇒ true)
      case Some(existing) ⇒
        db.run(allIntegrationTopics.filter(u ⇒ u.id == topic.id)
          .map(_.integrationTopicId).update(topic.integrationTopicId)).map(_ ⇒ false)
    }
  } else if (topic.integrationTopicId.isDefined) {
    find(topic.integrationId, topic.integrationGroupId, topic.integrationTopicId.get, top
      case None ⇒ db.run(allIntegrationTopics += topic).map(_ ⇒ true)
      case Some(existing) ⇒
        db.run(allIntegrationTopics.filter(u ⇒ u.integrationTopicId == topic.integratio
          u.integrationId == topic.integrationId && u.integrationGroupId == topic.integ
          .map(_.integrationTopicId).update(topic.integrationTopicId)).map(_ ⇒ false)
    }
  } else {
    throw new IllegalArgumentException
  }
}
```

End-User Environments

Professional End-User Environments

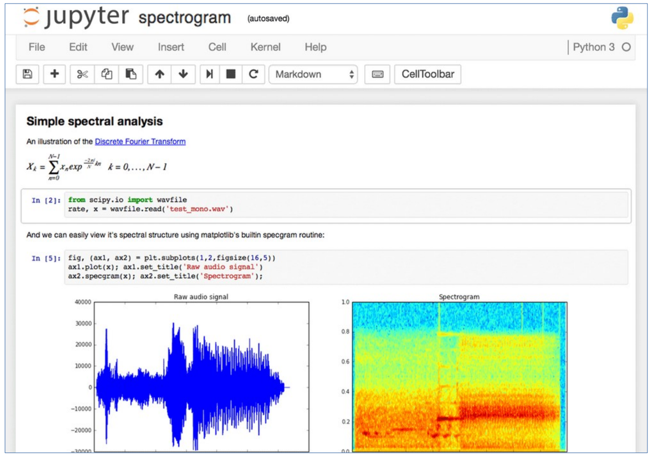
Professional Environments



=(B2/C2)*100

A	B	C	D	E
e	Score	Possible	Percentage	
	80	87	=(B2/C2)*100	
	53	87		
	51	87		
us	63	87		
	53	87		
ce	52	87		
	70	87		
pe	78	87		
e	81	87		
	54	87		

End-User Environments



Simple spectral analysis

An illustration of the Discrete Fourier Transform

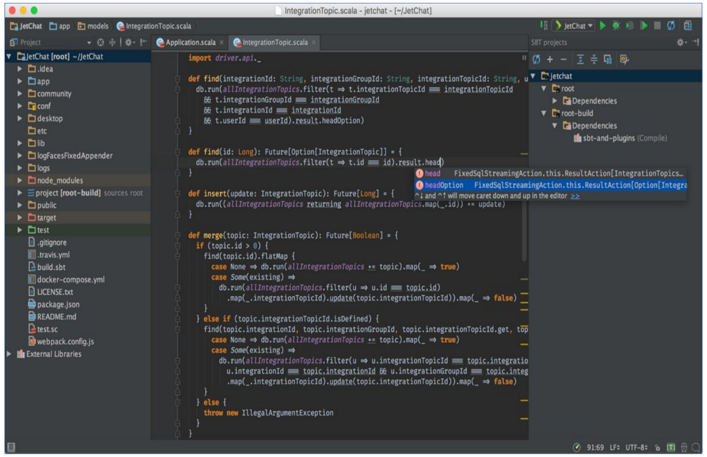
$$X_k = \sum_{n=0}^{N-1} x_n exp^{\frac{-2\pi i}{N}kn} \quad k = 0, \ldots, N-1$$

```
In [2]: from scipy.io import wavfile
        rate, x = wavfile.read('test_mono.wav')
```

And we can easily view it's spectral structure using matplotlib's builtin specgram routine:

```
In [5]: fig, (ax1, ax2) = plt.subplots(1,2,figsize(16,5))
        ax1.plot(x); ax1.set_title('Raw audio signal')
        ax2.specgram(x); ax2.set_title('Spectrogram');
```

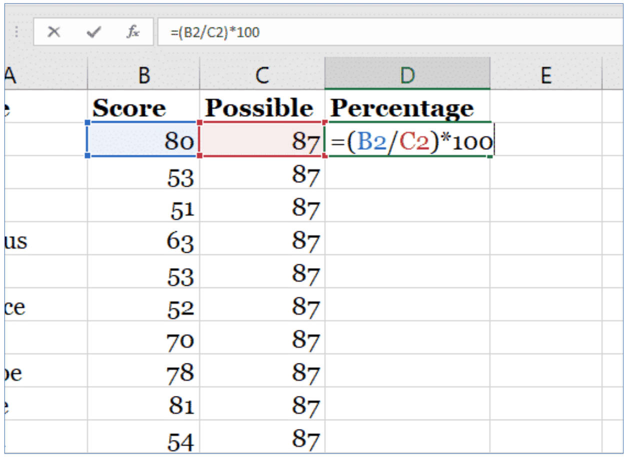
Professional End-User Environments



```
import driver.api._

def find(integrationId: String, integrationGroupId: String, integrationTopicId: String, u
  db.run(allIntegrationTopics.filter(t ⇒ t.integrationTopicId === integrationTopicId
    && t.integrationGroupId === integrationGroupId
    && t.integrationId === integrationId
    && t.userId === userId).result.headOption)
}

def find(id: Long): Future[Option[IntegrationTopic]] = {
  db.run(allIntegrationTopics.filter(t ⇒ t.id === id).result.head)
}                                                   head        FixedSqlStreamingAction.this.ResultAction[IntegrationTopics…
                                                    headOption  FixedSqlStreamingAction.this.ResultAction[Option[Integra
def insert(update: IntegrationTopic): Future[Long] = {   ^↓ and ^↑ will move caret down and up in the editor >>
  db.run((allIntegrationTopics returning allIntegrationTopics.map(_.id)) += update)
}

def merge(topic: IntegrationTopic): Future[Boolean] = {
  if (topic.id > 0) {
    find(topic.id).flatMap {
      case None ⇒ db.run(allIntegrationTopics += topic).map(_ ⇒ true)
      case Some(existing) ⇒
        db.run(allIntegrationTopics.filter(u ⇒ u.id === topic.id)
          .map(_.integrationTopicId).update(topic.integrationTopicId)).map(_ ⇒ false)
    }
  } else if (topic.integrationTopicId.isDefined) {
    find(topic.integrationId, topic.integrationGroupId, topic.integrationTopicId.get, top
      case None ⇒ db.run(allIntegrationTopics += topic).map(_ ⇒ true)
      case Some(existing) ⇒
        db.run(allIntegrationTopics.filter(u ⇒ u.integrationTopicId === topic.integratio
          u.integrationId === topic.integrationId && u.integrationGroupId === topic.integ
          .map(_.integrationTopicId).update(topic.integrationTopicId)).map(_ ⇒ false)
    }
  } else {
    throw new IllegalArgumentException
  }
}
```

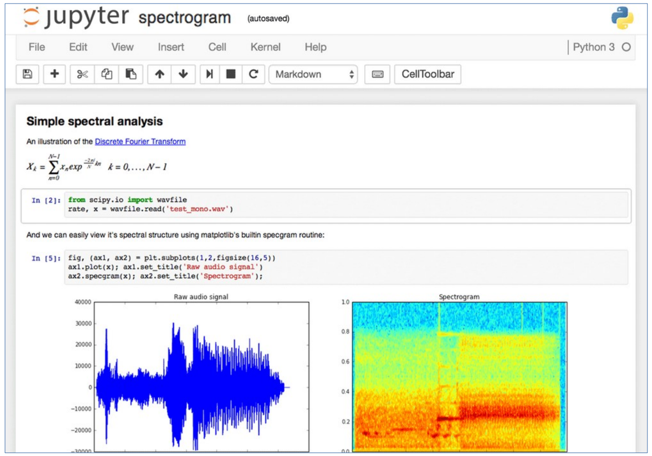
Professional Environments

- **Pure Functional PL**
- **Live Evaluation**
- **Direct Manipulation**



A	B	C	D	E
	Score	Possible	Percentage	
	80	87	=(B2/C2)*100	
	53	87		
	51	87		
us	63	87		
	53	87		
ce	52	87		
	70	87		
be	78	87		
	81	87		
	54	87		

End-User Environments



Simple spectral analysis

An illustration of the Discrete Fourier Transform

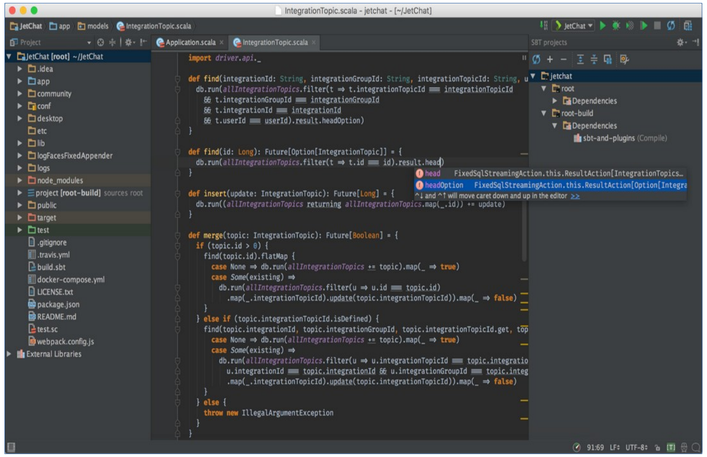
$$X_k = \sum_{n=0}^{N-1} x_n exp^{\frac{-2\pi i}{N}kn} \quad k = 0, \dots, N-1$$

```
In [2]: from scipy.io import wavfile
        rate, x = wavfile.read('test_mono.wav')
```

And we can easily view it's spectral structure using matplotlib's builtin specgram routine:

```
In [5]: fig, (ax1, ax2) = plt.subplots(1,2,figsize(16,5))
        ax1.plot(x); ax1.set_title('Raw audio signal')
        ax2.specgram(x); ax2.set_title('Spectrogram');
```

Professional End-User Environments

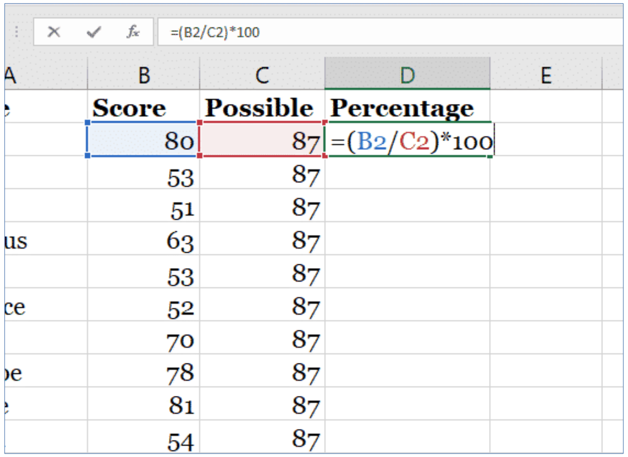


```
import driver.api._

def find(integrationId: String, integrationGroupId: String, integrationTopicId: String, u
  db.run(allIntegrationTopics.filter(t ⇒ t.integrationTopicId == integrationTopicId
    && t.integrationGroupId == integrationGroupId
    && t.integrationId == integrationId
    && t.userId == userId).result.headOption)
}

def find(id: Long): Future[Option[IntegrationTopic]] = {
  db.run(allIntegrationTopics.filter(t ⇒ t.id == id).result.head)
}
            head        FixedSqlStreamingAction.this.ResultAction[IntegrationTopics…
            headOption  FixedSqlStreamingAction.this.ResultAction[Option[Integra
            ^↓ and ^↑ will move caret down and up in the editor >>
def insert(update: IntegrationTopic): Future[Long] = {
  db.run((allIntegrationTopics returning allIntegrationTopics.map(_.id)) += update)
}

def merge(topic: IntegrationTopic): Future[Boolean] = {
  if (topic.id > 0) {
    find(topic.id).flatMap {
      case None ⇒ db.run(allIntegrationTopics += topic).map(_ ⇒ true)
      case Some(existing) ⇒
        db.run(allIntegrationTopics.filter(u ⇒ u.id == topic.id)
          .map(_.integrationTopicId).update(topic.integrationTopicId)).map(_ ⇒ false)
    }
  } else if (topic.integrationTopicId.isDefined) {
    find(topic.integrationId, topic.integrationGroupId, topic.integrationTopicId.get, top
      case None ⇒ db.run(allIntegrationTopics += topic).map(_ ⇒ true)
      case Some(existing) ⇒
        db.run(allIntegrationTopics.filter(u ⇒ u.integrationTopicId == topic.integratio
          u.integrationId == topic.integrationId && u.integrationGroupId == topic.integ
          .map(_.integrationTopicId).update(topic.integrationTopicId)).map(_ ⇒ false)
    }
  } else {
    throw new IllegalArgumentException
  }
}
```

Professional Environments

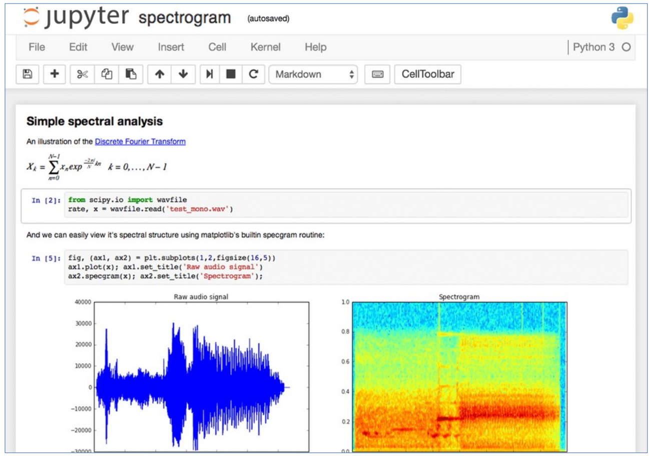
- Pure Functional PL
- Live Evaluation
- Direct Manipulation

- Static Typing
- Collaboration Facilities
- Automation



A	B	C	D	E
e	Score	Possible	Percentage	
	80	87	=(B2/C2)*100	
	53	87		
	51	87		
us	63	87		
	53	87		
ce	52	87		
	70	87		
pe	78	87		
e	81	87		
	54	87		

End-User Environments



Simple spectral analysis

An illustration of the Discrete Fourier Transform

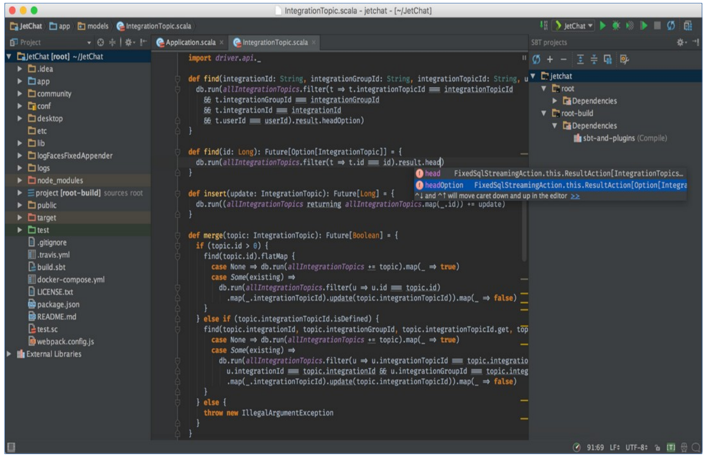
$$X_k = \sum_{n=0}^{N-1} x_n exp^{\frac{-2\pi i}{N}kn} \quad k = 0, \ldots, N-1$$

```
In [2]: from scipy.io import wavfile
        rate, x = wavfile.read('test_mono.wav')
```

And we can easily view it's spectral structure using matplotlib's builtin specgram routine:

```
In [5]: fig, (ax1, ax2) = plt.subplots(1,2,figsize(16,5))
        ax1.plot(x); ax1.set_title('Raw audio signal')
        ax2.specgram(x); ax2.set_title('Spectrogram');
```

Professional End-User Environments



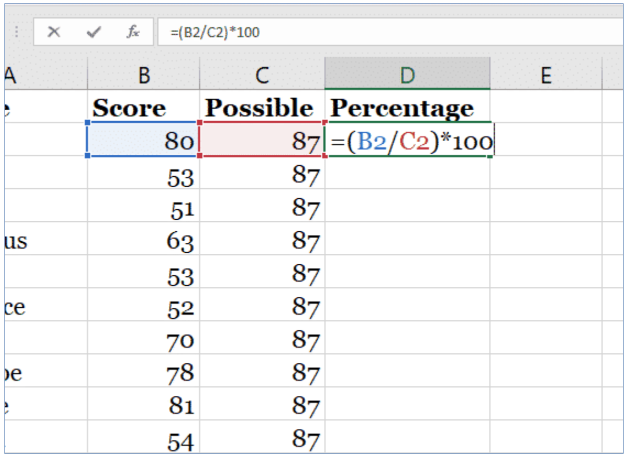
```
import driver.api._

def find(integrationId: String, integrationGroupId: String, integrationTopicId: String, u
  db.run(allIntegrationTopics.filter(t ⇒ t.integrationTopicId === integrationTopicId
    && t.integrationGroupId === integrationGroupId
    && t.integrationId === integrationId
    && t.userId === userId).result.headOption)
}

def find(id: Long): Future[Option[IntegrationTopic]] = {
  db.run(allIntegrationTopics.filter(t ⇒ t.id === id).result.head)
}
      head        FixedSqlStreamingAction.this.ResultAction[IntegrationTopics…
      headOption  FixedSqlStreamingAction.this.ResultAction[Option[Integra
      ^↓ and ^↑ will move caret down and up in the editor >>

def insert(update: IntegrationTopic): Future[Long] = {
  db.run((allIntegrationTopics returning allIntegrationTopics.map(_.id)) += update)
}

def merge(topic: IntegrationTopic): Future[Boolean] = {
  if (topic.id > 0) {
    find(topic.id).flatMap {
      case None ⇒ db.run(allIntegrationTopics += topic).map(_ ⇒ true)
      case Some(existing) ⇒
        db.run(allIntegrationTopics.filter(u ⇒ u.id === topic.id)
          .map(_.integrationTopicId).update(topic.integrationTopicId)).map(_ ⇒ false)
    }
  } else if (topic.integrationTopicId.isDefined) {
    find(topic.integrationId, topic.integrationGroupId, topic.integrationTopicId.get, top
      case None ⇒ db.run(allIntegrationTopics += topic).map(_ ⇒ true)
      case Some(existing) ⇒
        db.run(allIntegrationTopics.filter(u ⇒ u.integrationTopicId === topic.integratio
          u.integrationId === topic.integrationId && u.integrationGroupId === topic.integ
          .map(_.integrationTopicId).update(topic.integrationTopicId)).map(_ ⇒ false)
    }
  } else {
    throw new IllegalArgumentException
  }
}
```

Professional Environments

- **Pure Functional PL**
- **Live Evaluation**
- **Direct Manipulation**

- **Dynamic Typing**
- **Programmable Documents**

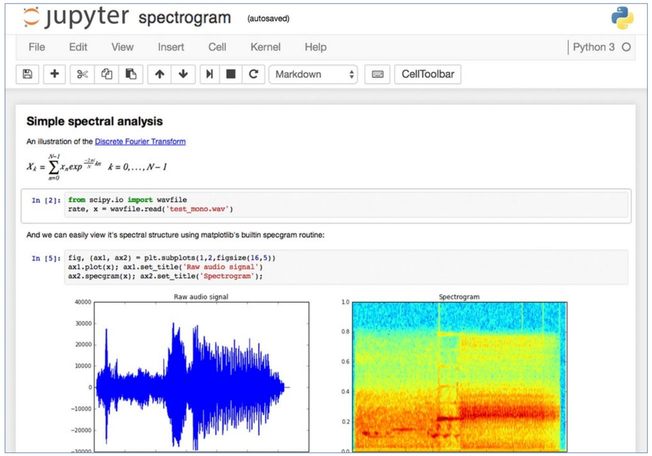
- **Static Typing**
- **Collaboration Facilities**
- **Automation**



=(B2/C2)*100

A	B	C	D	E
e	Score	Possible	Percentage	
	80	87	=(B2/C2)*100	
	53	87		
	51	87		
us	63	87		
	53	87		
ce	52	87		
	70	87		
pe	78	87		
e	81	87		
	54	87		

End-User Environments


jupyter spectrogram (autosaved)
File Edit View Insert Cell Kernel Help
Python 3
Markdown
CellToolbar
Raw audio signal
Spectrogram

Simple spectral analysis

An illustration of the Discrete Fourier Transform

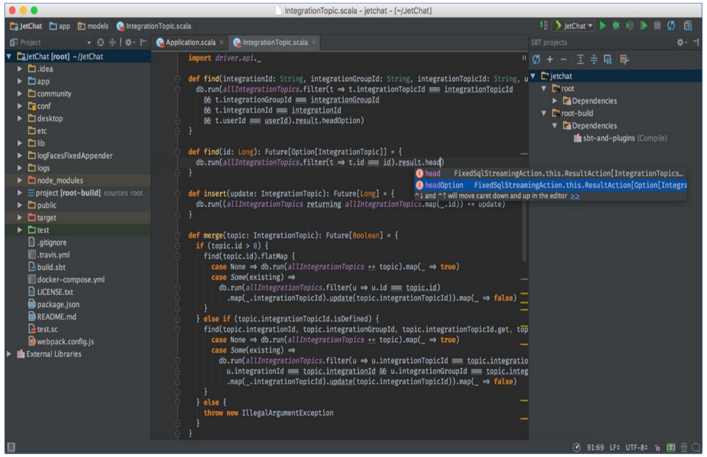
$$X_k = \sum_{n=0}^{N-1} x_n exp^{\frac{-2\pi i}{N} kn} \quad k = 0, \ldots, N-1$$

```
In [2]: from scipy.io import wavfile
        rate, x = wavfile.read('test_mono.wav')
```

And we can easily view it's spectral structure using matplotlib's builtin specgram routine:

```
In [5]: fig, (ax1, ax2) = plt.subplots(1,2,figsize(16,5))
        ax1.plot(x); ax1.set_title('Raw audio signal')
        ax2.specgram(x); ax2.set_title('Spectrogram');
```

Professional End-User Environments



```
import driver.api._

def find(integrationId: String, integrationGroupId: String, integrationTopicId: String, u
  db.run(allIntegrationTopics.filter(t ⇒ t.integrationTopicId == integrationTopicId
    && t.integrationGroupId == integrationGroupId
    && t.integrationId == integrationId
    && t.userId == userId).result.headOption)
}

def find(id: Long): Future[Option[IntegrationTopic]] = {
  db.run(allIntegrationTopics.filter(t ⇒ t.id == id).result.head)
}

def insert(update: IntegrationTopic): Future[Long] = {
  db.run((allIntegrationTopics returning allIntegrationTopics.map(_.id)) += update)
}

def merge(topic: IntegrationTopic): Future[Boolean] = {
  if (topic.id > 0) {
    find(topic.id).flatMap {
      case None ⇒ db.run(allIntegrationTopics += topic).map(_ ⇒ true)
      case Some(existing) ⇒
        db.run(allIntegrationTopics.filter(u ⇒ u.id == topic.id)
          .map(_.integrationTopicId).update(topic.integrationTopicId)).map(_ ⇒ false)
    }
  } else if (topic.integrationTopicId.isDefined) {
    find(topic.integrationId, topic.integrationGroupId, topic.integrationTopicId.get, top
      case None ⇒ db.run(allIntegrationTopics += topic).map(_ ⇒ true)
      case Some(existing) ⇒
        db.run(allIntegrationTopics.filter(u ⇒ u.integrationTopicId == topic.integratio
          u.integrationId == topic.integrationId && u.integrationGroupId == topic.integ
          .map(_.integrationTopicId).update(topic.integrationTopicId)).map(_ ⇒ false)
    }
  } else {
    throw new IllegalArgumentException
  }
}
```

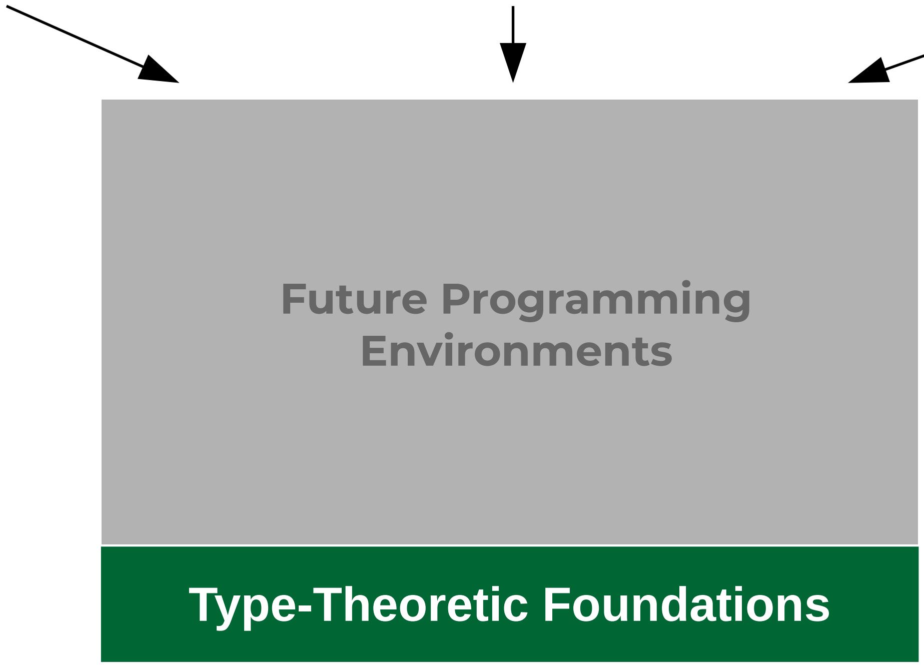
```
head        FixedSqlStreamingAction.this.ResultAction[IntegrationTopics…
headOption  FixedSqlStreamingAction.this.ResultAction[Option[Integra
^↓ and ^↑ will move caret down and up in the editor >>
```

Professional Environments

- **Pure Functional PL**
- **Live Evaluation**
- **Direct Manipulation**

- **Dynamic Typing**
- **Programmable Documents**

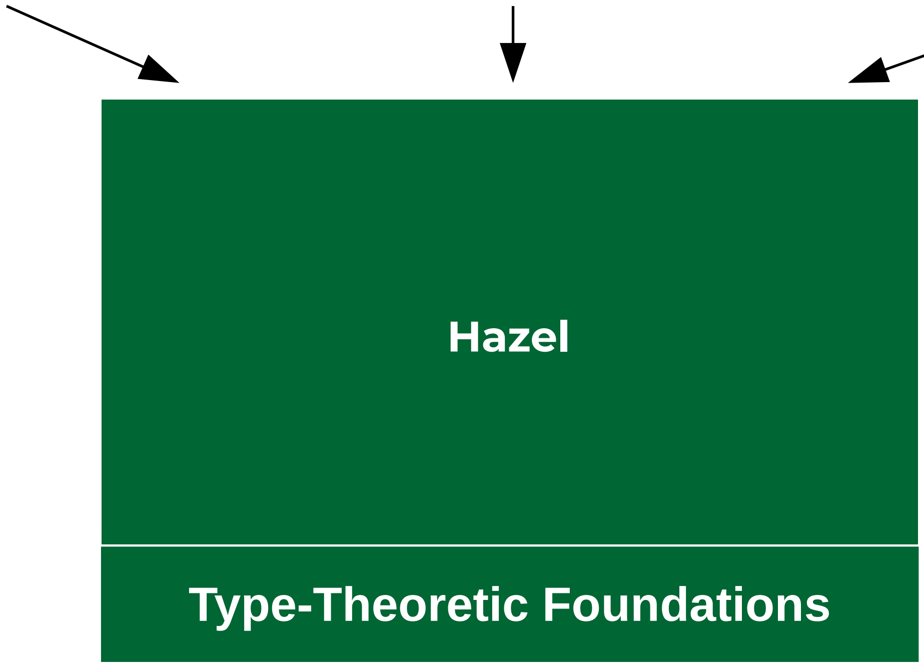
- **Static Typing**
- **Collaboration Facilities**
- **Automation**



- **Pure Functional PL**
- **Live Evaluation**
- **Direct Manipulation**

- **Dynamic Typing**
- **Programmable Documents**

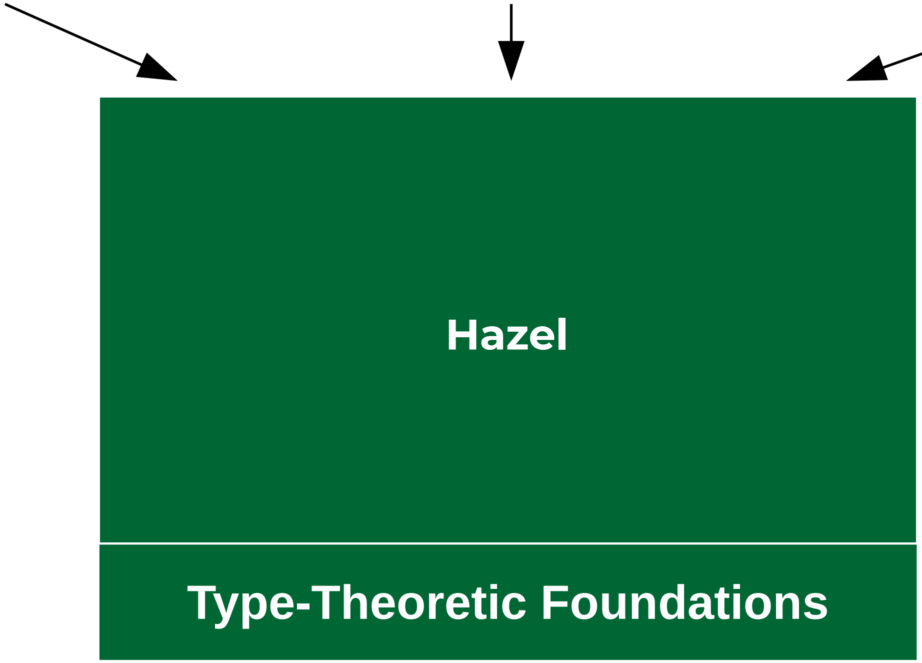
- **Static Typing**
- **Collaboration Facilities**
- **Automation**

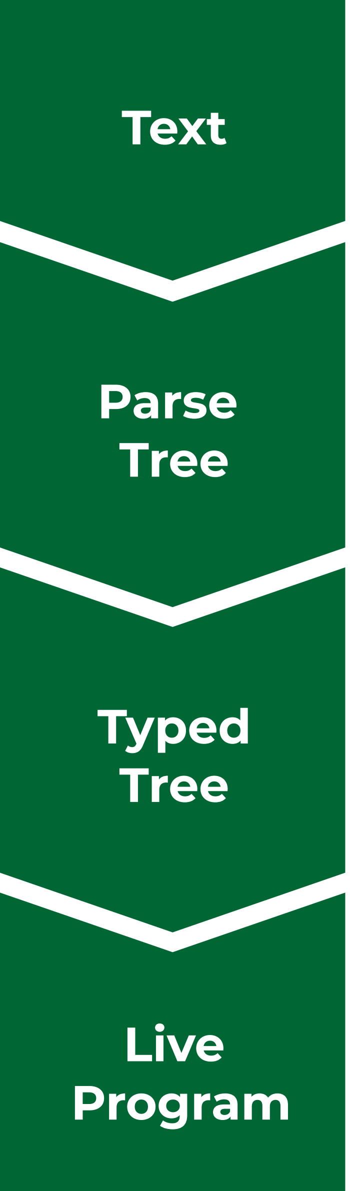


- Pure Functional PL
- Live Evaluation
- Direct Manipulation

- Dynamic Typing
- Programmable Documents

- Static Typing
- Collaboration Facilities
- Automation





The **gap problem**.

Text

Parse Tree

Typed Tree

Live Program

The **gap problem**.

Text

Parse Tree

Typed Tree

Live Program

è""55pooooodffffffffffffffffff
fffffffffffffffffffffffffff'fd""'ddd

Hazel **solves** the gap problem using **typed holes**.

Every editor state in Hazel is semantically meaningful.
(It has a type, it has a result, and it can be transformed as a tree.)

[Omar et al., POPL 2017] & [Omar et al., POPL 2019]

Demo: Live Programming with Typed Holes in Hazel

See hazel.org

Hazel **solves** the gap problem using **typed holes**.

Every editor state in Hazel is semantically meaningful.
(It has a **type**, it has a **result**, and it **can be transformed** as a tree.)

[Omar et al., POPL 2017] & [Omar et al., POPL 2019]

Ongoing Work: Hazel Assistant

Let’s use types + examples + live evaluation + edit action history + statistics to fill holes (i.e. synthesis within program sketches).

Ongoing Work: Hazel Assistant

Let’s use **types** + **examples** + live evaluation + edit action history + statistics to fill one hole.

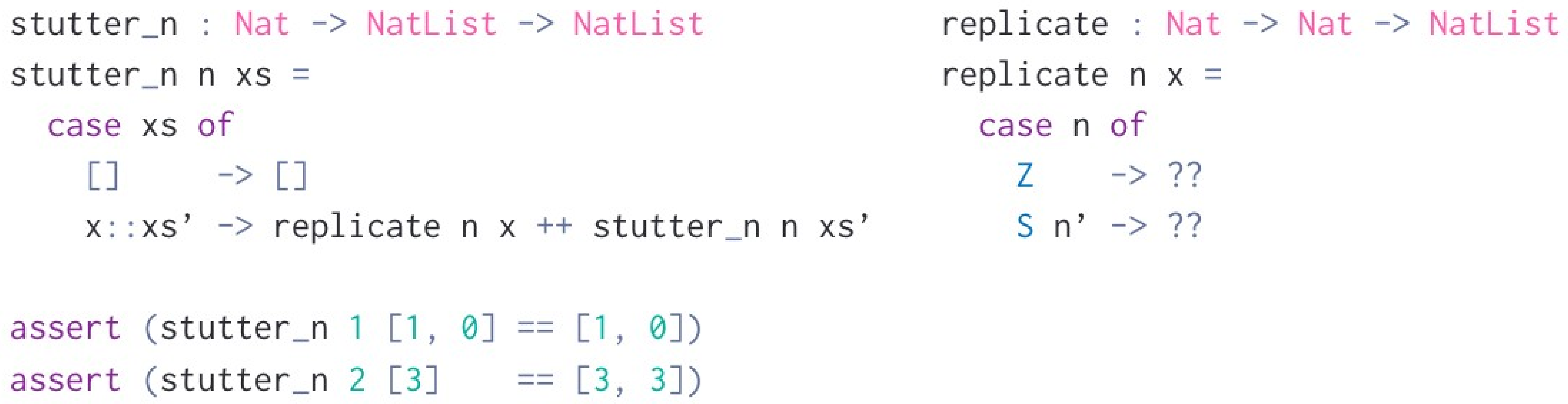
[Osera and Zdancewic, PLDI 2015]

Ongoing Work: Hazel Assistant

Let's use **types** + **examples** + **live evaluation** + edit action history + statistics to fill holes (i.e. synthesis within program sketches).

[Lubin, Collins, Omar, and Chugh, *Program Sketching with Live Bidirectional Evaluation*, ICFP 2020] + https://uchicago-pl.github.io/smyth/

Ongoing Work: Hazel Assistant



```
stutter_n : Nat -> NatList -> NatList
stutter_n n xs =
  case xs of
    []     -> []
    x::xs' -> replicate n x ++ stutter_n n xs'

assert (stutter_n 1 [1, 0] == [1, 0])
assert (stutter_n 2 [3]    == [3, 3])
```

```
replicate : Nat -> Nat -> NatList
replicate n x =
  case n of
    Z    -> ??
    S n' -> ??
```

Fig. 1. A program sketch in Smyth to "stutter" each element of a list n times. The desired solutions for the holes in `replicate` are `[]` for the `Z` branch and `x :: replicate n' x` for the `S` branch.

[Lubin, Collins, Omar, and Chugh, ICFP 2020]

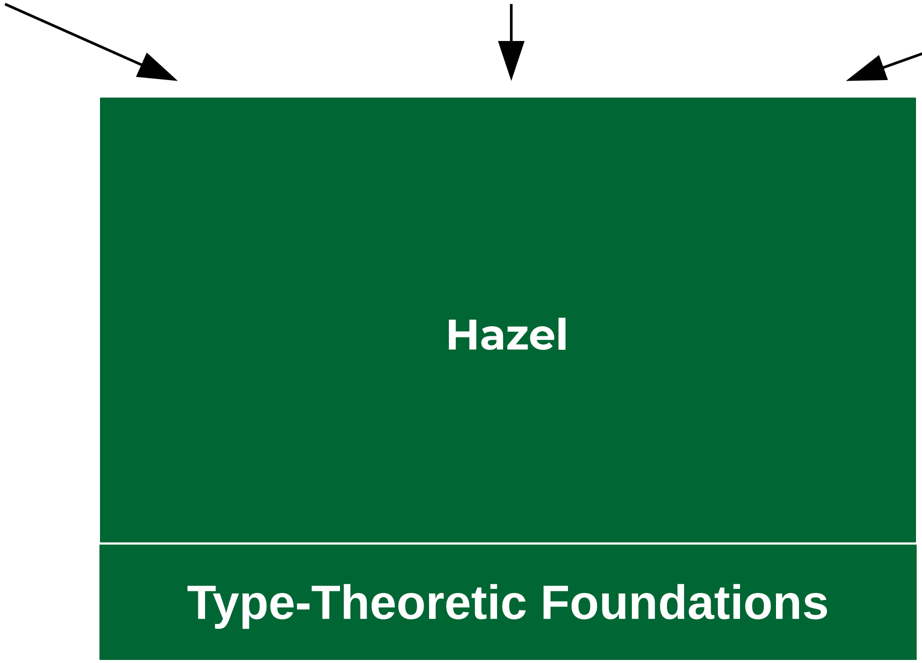
Ongoing Work: Hazel Assistant

Let's use **types** + **examples** + **live evaluation** + **edit action history** + **statistics** to fill holes (i.e. synthesis within program sketches).

- Pure Functional PL
- Live Evaluation
- Direct Manipulation

- Dynamic Typing
- Programmable Documents

- Static Typing
- Collaboration Facilities
- Automation


Hazel
Type-Theoretic Foundations