Hedy: a gradual programming language

Jesús Pelay
What is Hedy?

1. Gradual
2. Multilingual
3. Built for teaching
Misconceptions

1. Compilers are friends!
2. Syntax is not really an issue
3. You mostly learn alone!
Can you teach us programming?

Of course!

Felienne Hermans
Scratch
Scratch

But teacher we don’t want a toy anymore!

We want textual languages!
1 print("Hello world!")

Results of your code will appear here when you run the project.
Python

```python
print("Hello world!")
```

```
Hello world!
```
Python

```python
print("Hello world!")
```

Results of your code will appear here when you run the project.
```python
1  Print("Hello world!")
```

Traceback (most recent call last):
  File "/home/runner/HedyTalk/main.py", line 1, in <module>
    Print("Hello world!")
NameError: name 'Print' is not defined. Did you mean: 'print'?
```python
1    print("Hello world!")
```

```
File "/home/runner/HedyTalk/main.py", line 1
      print("Hello world!")
    ^
SyntaxError: ' was never closed
```
```python
1  print("Hello world!")
```

Results of your code will appear here when you Run the project.
```python
print("Hello world!")
```
Python

```
1  for i in range(4):
2      print(i)
```
Python

```
for i in range(4):
    print(i)
```

Syntax creates cognitive overload.

Colon

Brackets

Spaces
Cognitive Overload

a in e

- cat in tree
- Cat in tree
- Cat in tree.
- The cat in the tree.
Cognitive Overload

- a in e
- cat in tree
- Cat in tree
- Cat in tree.
- The cat in the tree.
Cognitive Overload

- a in e
- cat in tree
- Cat in tree
- Cat in tree.
- The cat in the tree.
Cognitive Overload

The cat in the tree.

Cat in tree.

Cat in tree.
Cognitive Overload

The cat in the tree.

Cat in tree.

Cat in tree.

Cat in tree.
Cognitive Overload

Rules change gradually!

\[
\begin{align*}
\text{a in e} \\
cat in tree \\
Cat in tree \\
Cat in tree.
\end{align*}
\]
Cognitive Overload

Rules change gradually in math too!

5-3 = 2
3-5 = 0
Cognitive Overload

Rules change gradually in math too!

\[ \begin{align*}
5-3 &= 2 \\
3-5 &= 0 \\
3-5 &= -2 \\
\end{align*} \]
Cognitive Overload

Rules change gradually in math too!

5 - 3 = 2
3 - 5 = 0
3 - 5 = -2
8 / 3 = 2r2
8 / 3 = 2\frac{2}{3}
8 / 3 = 2.666
Cognitive Overload

Let’s teach code gradually as well!
Cognitive Overload

Let’s teach code gradually as well!
print Hello
ask What's your name
eco hello

print 'Hello'
name is ask 'What is your name?'
print 'Your name is ' name ' so pretty!'

print('Hello')
name = input('What is your name=')
print('Your name is ', name, ' so pretty!')
Design Goals

1. Concepts are offered at least three times.
2. The concept is introduced as simple as possible.
3. Only one aspect changes at a time.
4. Syntactic elements are deferred to the latest moment.
5. Concepts are interleaved.
6. It’s always possible to create meaningful programs.
What is Hedy?

1. Gradual
2. Multilingual
3. Built for teaching
What is Hedy?

1. Gradual
2. Multilingual
3. Built for teaching
This is great but we’d like to program in Dutch!

Ok!
Multilingual

Hedy is available in 47 languages!
Multilingual

Demo time

1. `imprimir` ¡Hola, programador!
2. 
3. 

1. قولنا البيضاء هدي
2. 
3. 

1. `かげ` オウムのヘディーです。
2. 
3. 

1. `drucke` Was wählst du?
2. 
3. 

What is Hedy?

1. Gradual
2. Multilingual
3. Built for teaching
What is Hedy?

1. Gradual
2. Multilingual
3. Built for teaching
Built for teaching

Easier to teach!

The levels are like a step-by-step guide
Built for teaching
Built for teaching

What can my students build?
Built for teaching

Lesson!
Built for teaching

What do I do now?
Built for teaching

I built this! Can you help me debug it?
Built for teaching
Adventures!
Classes!
Quizzes!
Built for teaching

Customizable too!
Architecture

Client
- Skulpt
- Ace

Server
- Lark
- Flask
Architectural Challenges

Client
- Skulpt
- Ace

Server
- Lark
- Flask

Doesn’t work well with RTL languages
We replaced it! It was hard!
Architecture Challenges

Parser generators are not made for gradual multilingual languages!

```plaintext
program: _EOL* (command _EOL+)*
command?
command: print | ask | echo | error
print: _PRINT (text)?
ask: _ASK (text)?
echo: _ECHO (text)?
```

```plaintext
@top Program { eol* (Command eol+)* Command? }
Command {
    Print | Ask | Echo | Play | Turtle | ErrorInvalid
}
Print { print+ Text+ }
Play { play+ Text+ }
Ask { ask+ Text+ }
Echo { echo+ Text* }
```
Open Source
Open Source
Open Source

Come join us!

github.com/hedyorg/hedy

hedy.org