

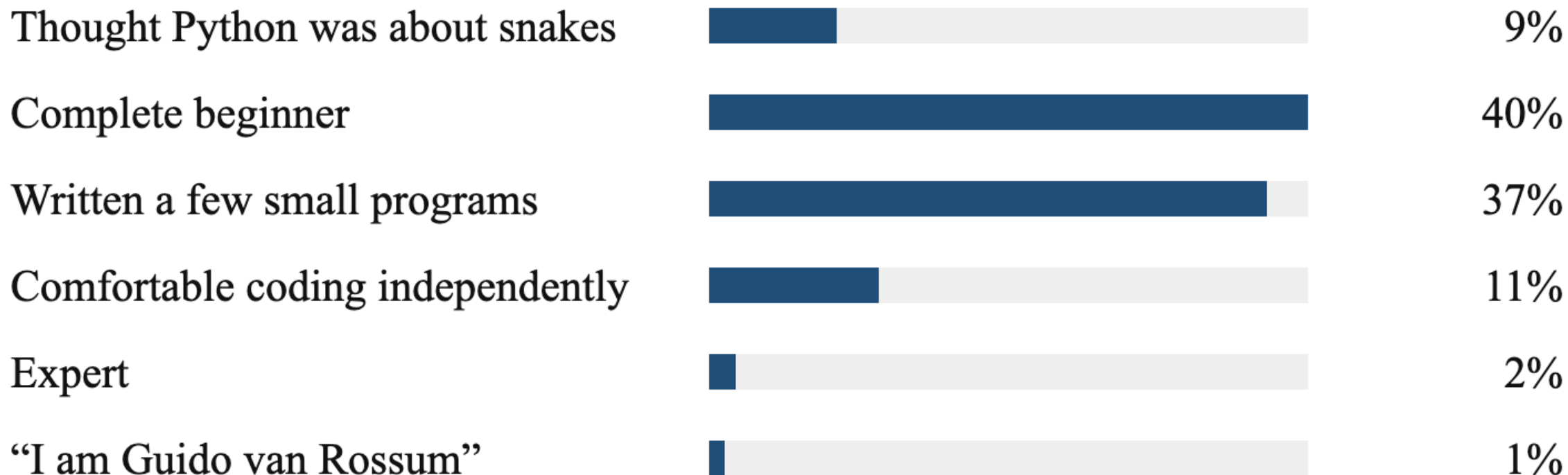
L-2

Python Fundamentals: Variables, Operators and Expressions

ENGG1330
Computer Programming I

We're Starting at Different Levels

49% selected one of the two beginner options



How Did Learning to Code Feel?

“Excited” was the most common response

22%

excited

5%

interesting

4%

exciting

4%

nervous

3%

fun

3%

hard

2%

anxious

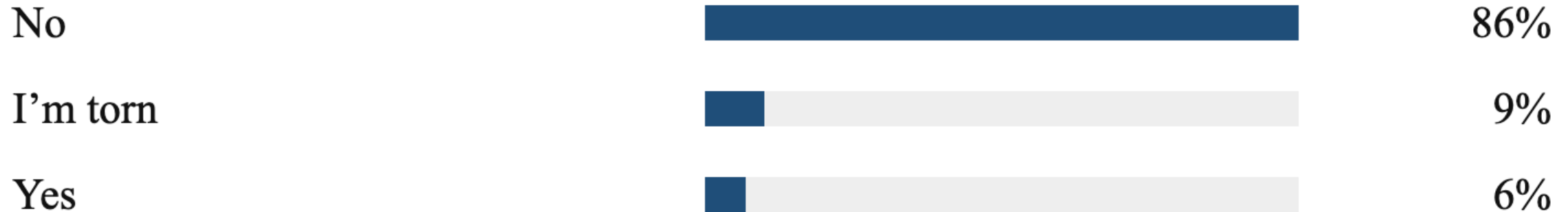
2%

curious

Would You Press the Button?

A guaranteed B · No work · Nothing learned

86% chose “No”



L-1 Participation Snapshot

Approximate figures reported by Kevin



The survey is only one part: remember the coding challenges too

Terminology

ENGG1330
Computer Programming 1

Terminology

We combine *literals* and *variables* with *operators*
to form *expressions*

Terminology

We combine *literals* and *variables* with *operators*
to form *expressions*



Variables?
Operators?
Literals?
Expressions?

Terminology



```
PY main.py
1 pi = 3.14159
2 diameter = 3
3 area = pi * ( ( diameter / 2 ) ** 2 )
4 print(area)
```

Download: engg1330.dirk.hk/lec_02_circle_area.py

Terminology



```
PY main.py
1 3.14159
2      3
3          2      2
4
```

Download: engg1330.dirk.hk/lec_02_circle_area.py

Literals

Raw data in
code

Terminology



```
PY main.py
1 pi
2 diameter
3 area pi diameter
4
```

Download: engg1330.dirk.hk/lec_02_circle_area.py

Variables

Remember stuff
by storing it in
memory

Terminology



```
PY main.py
1  =
2      =
3  = * ( ( / ) ** )
4
Download: engg1330.dirk.hk/lec\_02\_circle\_area.py
```

Operators

Special symbols
that carry out
operations

Terminology



```
PY main.py
1 | 3.14159
2 |
3 |     pi * ( ( diameter / 2 ) ** 2 )
4 |
Download: engg1330.dirk.hk/lec\_02\_circle\_area.py
```

Expressions

Combine
literals,
variables and
operators

Terminology

Literals

Raw data in code

Variables

Store data in memory

Operators

Special symbols that carry out operations

Expressions

Combine literals and variables to create new values

Literals

ENGG1330
Computer Programming 1

Literals Have Different Data Types

```
PY main.py
1 3.14159
2      3
3          2      2
4
```

Download: engg1330.dirk.hk/lec_02_circle_area.py

Literals Have Different Data Types

```
PY main.py
1 3.14159
2      3
3          2      2
4
```

Download: engg1330.dirk.hk/lec_02_circle_area.py

3
Integer (int)

3.14159
Float (float)

'Bruce Lee'
String (str)

True
Boolean (bool)

Variables

ENGG1330
Computer Programming 1

Variables Are Used to Store Data in Memory

```
PY main.py
1 a = 3
2 b = 3.14159
3 c = 'Bruce Lee'
4 d = True
```

Download: engg1330.dirk.hk/lec_02_example_17_02.py

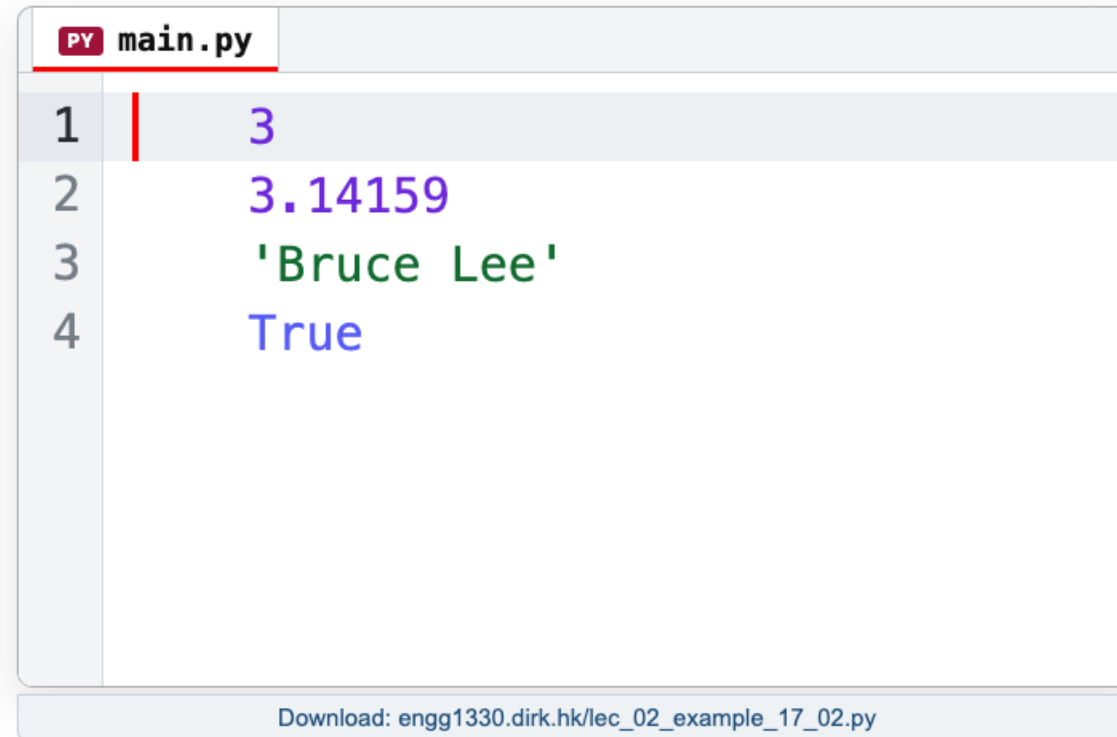
Variables Have Unique Names (Identifiers)



```
PY main.py
1 a
2 b
3 c
4 d
```

Download: engg1330.dirk.hk/lec_02_example_17_02.py

Variables Have a Value



```
PY main.py
1 3
2 3.14159
3 'Bruce Lee'
4 True
```

Download: engg1330.dirk.hk/lec_02_example_17_02.py

Variables Have a Type

```
PY main.py
1 a = 3
2 b = 3.14159
3 c = 'Bruce Lee'
4 d = True
```

Download: engg1330.dirk.hk/lec_02_example_17_02.py

Use `type(x)` to Determine the Type of the Variable `x`

```
PY main.py
1 a = 3
2 b = 3.14159
3 c = 'Bruce Lee'
4 d = True
5 print(type(a))
6 print(type(b))
7 print(type(c))
8 print(type(d))
```

Download: engg1330.dirk.hk/lec_02_example_21_03.py

You Can Also Directly Write the Literal in type(x)

```
PY main.py
1 print(type(3))
2 print(type(3.14159))
3 print(type('Bruce Lee'))
4 print(type(True))
```

Download: engg1330.dirk.hk/lec_02_example_22_04.py

One Name Can Refer to Values of Different Types

```
PY main.py
1 | age = 18
2 | print(type(age))      # int
3 |
4 | age = "18"
5 | print(type(age))      # str
```

Download: engg1330.dirk.hk/lec_02_value_types.py

Type Hints



```
1 age: int = 18
2 print(type(age))
```

Download: engg1330.dirk.hk/lec_02_type_hints.py

: `int` documents the intended type

`int(...)` converts a value

Python does not enforce type hints at runtime

A Variable Is Like a Blackboard

```
PY main.py
1 age = 18
2 print('You are', age)
3 age = age + 1
4 print('You will be', age, 'soon')
```

Download: engg1330.dirk.hk/lec_02_example_23_05.py



[Image source](#)

Names Describe What a Variable Is Used For



`d = 7`



`elapsedTimeInDays = 7`

Choosing Variable Names

Choose meaningful names that describe the value

```
elapsed_days = 7
```

```
student_count = 120
```

```
total_cost = 85.50
```

Start with a letter or underscore

Then use letters, digits or underscores

Valid and Invalid Names

Invalid	Reason	Valid Alternative
2nd_score	Starts with a digit	second_score
class	Reserved keyword	class_name
total-cost	Contains a hyphen	total_cost

Names are case sensitive

age and **Age** are different names

Identifiers Are Case Sensitive

```
PY main.py
1 X = 'awesome'
2 print('Python is',x)
```

Download: engg1330.dirk.hk/lec_02_example_32_07.py

```
PY main.py
1 x = 'awesome'
2 print('Python is',x)
```

Download: engg1330.dirk.hk/lec_02_example_32_08.py

Operators: Arithmetic

Arithmetic Operators

Operator	Name	Description	Example	Result
+	Addition	Adds the operands	1 + 1	2
-	Subtraction	Subtracts the right operand	5 - 4	1
*	Multiplication	Multiplies the operands	3 * 7	21
/	Division	Divides the left by the right	3 / 6	0.5
%	Modulo	Returns the division remainder	7 % 3	1
**	Exponent	Raises the left to a power	7 ** 3	343
//	Floor Division	Rounds the quotient down	7 // 3	2

Arithmetic Operators: Addition

PY main.py

```
1 numCookiesToday = 3
2 numCookiesYesterday = 4
3 print('You had', numCookiesYesterday + numCookiesToday, 'total!')
```

Download: engg1330.dirk.hk/lec_02_example_37_09.py



Arithmetic Operators: Division

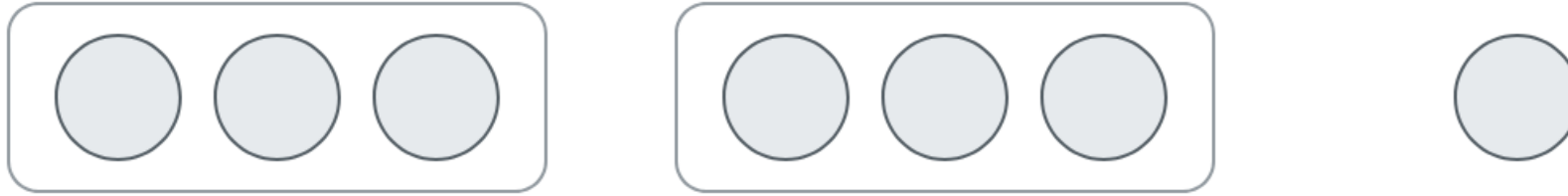
```
PY main.py
1 a = 42
2 b = 0
3 print(a/b)
```

Download: engg1330.dirk.hk/lec_02_example_38_10.py

Modulus vs. Floor Division

Modulo: What Is Left Over?

7 objects, groups of 3

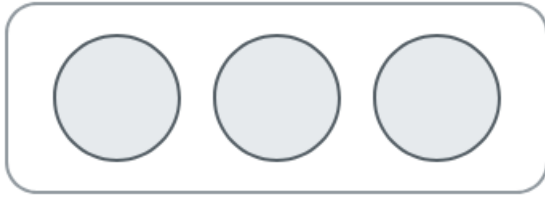
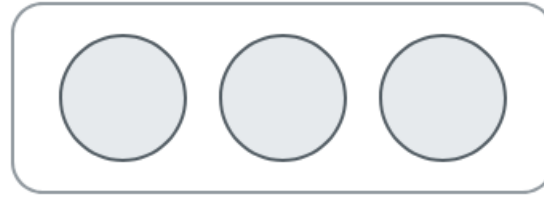


```
PY main.py
1 print(7 % 3)
```

Download: engg1330.dirk.hk/lec_02_modulo_groups.py

Modulo: What Is Left Over?

7 objects, groups of 3



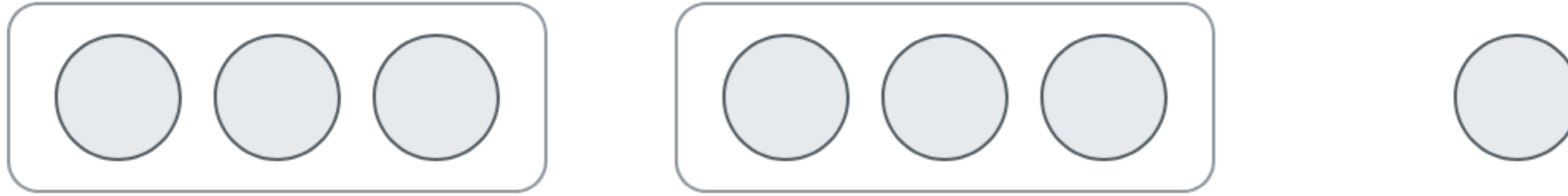
1 left over

```
PY main.py
1 print(7 % 3)
```

Download: engg1330.dirk.hk/lec_02_modulo_groups.py

Floor Division: How Many Complete Groups?

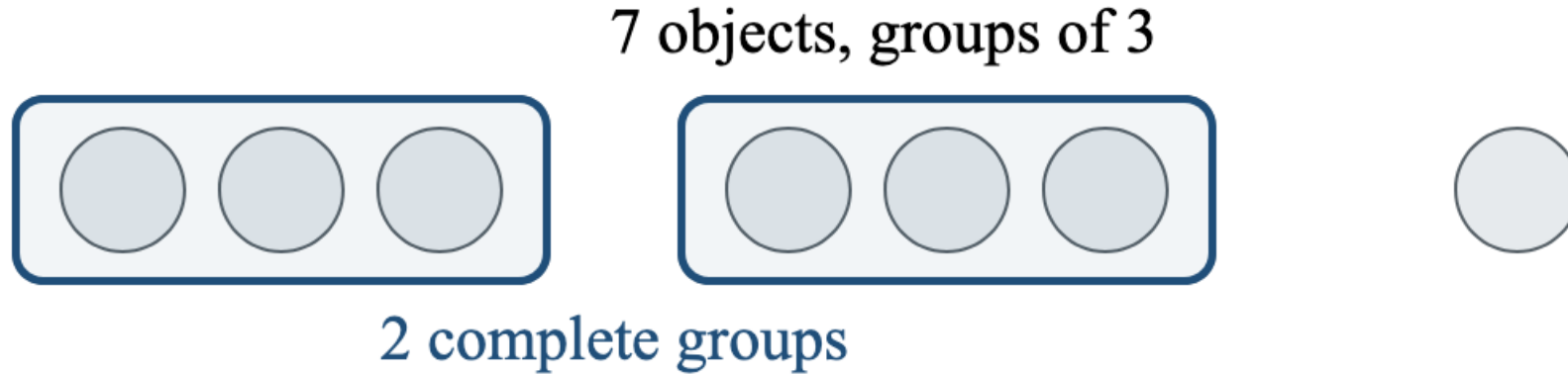
7 objects, groups of 3



```
PY main.py
1 print(7 // 3)
```

Download: engg1330.dirk.hk/lec_02_floor_groups.py

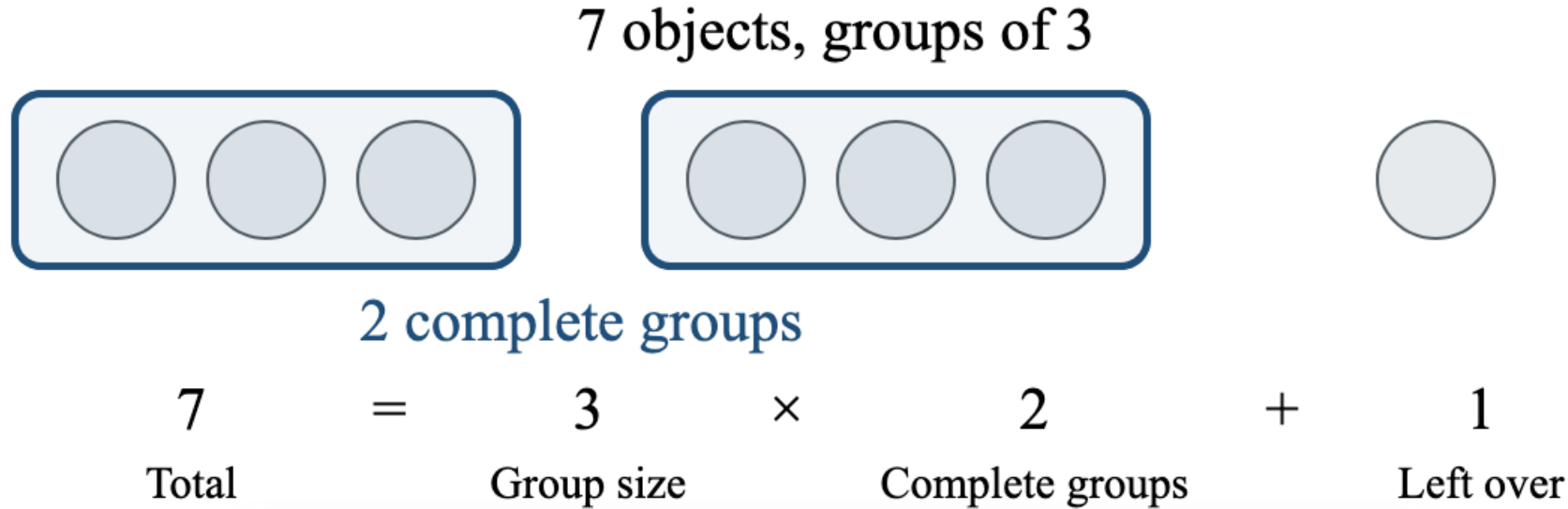
Floor Division: How Many Complete Groups?



```
PY main.py
1 print(7 // 3)
```

Download: engg1330.dirk.hk/lec_02_floor_groups.py

Floor Division: How Many Complete Groups?



```
PY main.py
1 print(7 // 3)
```

Download: engg1330.dirk.hk/lec_02_floor_groups.py

Arithmetic Operators: Modulus

```
PY main.py
1 a = 42
2 b = 0
3 print(a%b)
```

Download: engg1330.dirk.hk/lec_02_example_42_11.py

Modulo Checks Whether a Number Is Even or Odd

Numbers	Examples	Test
Even	2, 4, 6, 8, ...	$x \% 2 == 0$
Odd	1, 3, 5, 7, ...	$x \% 2 == 1$

Operators: Assignment

Assignment Operator



```
PY main.py
1 a = 5
2 print(a)
```

Download: engg1330.dirk.hk/lec_02_example_46_12.py

Compound Assignment Operator

```
PY main.py
1 a = 5
2 a += 5
3 print(a)
```

Download: engg1330.dirk.hk/lec_02_example_49_14.py

L-2 Survey

Complete the full survey during the lecture

Operators: Boolean and Comparisons (If / Else)

Comparisons Produce Boolean Values

PY main.py

```
1 x = 42
2 xMod2 = x % 2
3 even = xMod2 == 0
4 print(x, 'is even:', even)
```

Download: engg1330.dirk.hk/lec_02_example_52_16.py

PY main.py

```
1 x = 43
2 xMod2 = x % 2
3 even = xMod2 == 0
4 print(x, 'is even:', even)
```

Download: engg1330.dirk.hk/lec_02_example_52_17.py

if Runs Code When Its Condition Is True

PY main.py

```
1 x = 42
2 xMod2 = x % 2
3 even = xMod2 == 0
4 if even:
5     print(x, 'is even:', even)
```

Download: engg1330.dirk.hk/lec_02_example_53_18.py

PY main.py

```
1 x = 43
2 xMod2 = x % 2
3 even = xMod2 == 0
4 if even:
5     print(x, 'is even:', even)
```

Download: engg1330.dirk.hk/lec_02_example_53_19.py

Branching with if and else

PY main.py

```
1 x = True
2 if (x):
3     print('yes')
4 else:
5     print('no')
```

Download: engg1330.dirk.hk/lec_02_example_54_20.py

PY main.py

```
1 x = False
2 if (x):
3     print('yes')
4 else:
5     print('no')
```

Download: engg1330.dirk.hk/lec_02_example_54_21.py

Indentation Marks Blocks of Code

```
main.py
1 x = True
2 if (x):
3     print('yes')
4 else:
5     print('no')
```

Download: engg1330.dirk.hk/lec_02_example_54_20.py

← A block of code

← Another block of code

Comparison Operators with if and else

PY main.py

```
1 x = 42
2 xMod2 = x % 2
3 if (xMod2 == 0):
4     print(x, 'is even')
5 else:
6     print(x, 'is odd')
```

Download: engg1330.dirk.hk/lec_02_example_56_22.py

PY main.py

```
1 x = 43
2 xMod2 = x % 2
3 if (xMod2 == 0):
4     print(x, 'is even')
5 else:
6     print(x, 'is odd')
```

Download: engg1330.dirk.hk/lec_02_example_56_23.py

Greater Than or Equal To

PY main.py

```
1 age = 17
2 if (age >= 18):
3     print('You are an adult')
4 else:
5     print('You are under age')
```

Download: engg1330.dirk.hk/lec_02_example_57_24.py

PY main.py

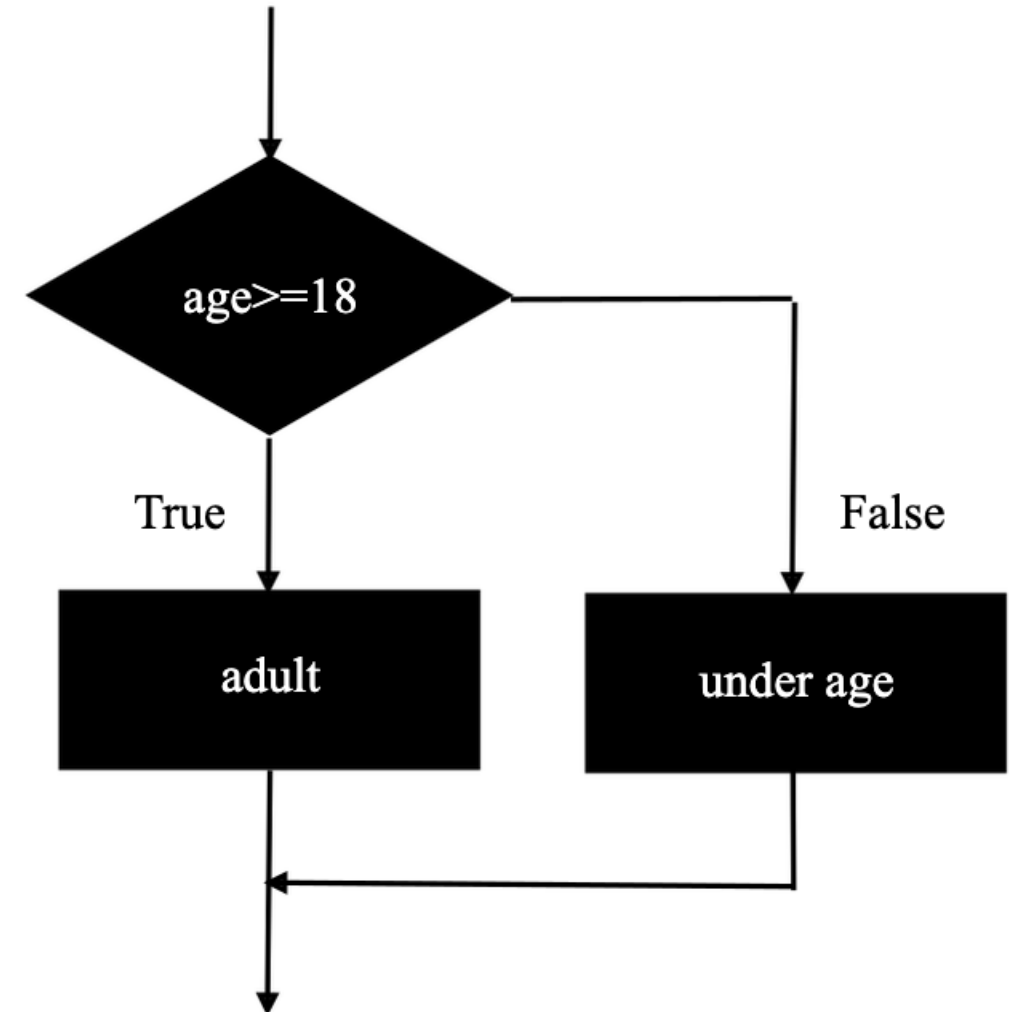
```
1 age = 36
2 if (age >= 18):
3     print('You are an adult')
4 else:
5     print('You are under age')
```

Download: engg1330.dirk.hk/lec_02_example_57_25.py

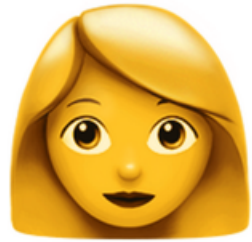
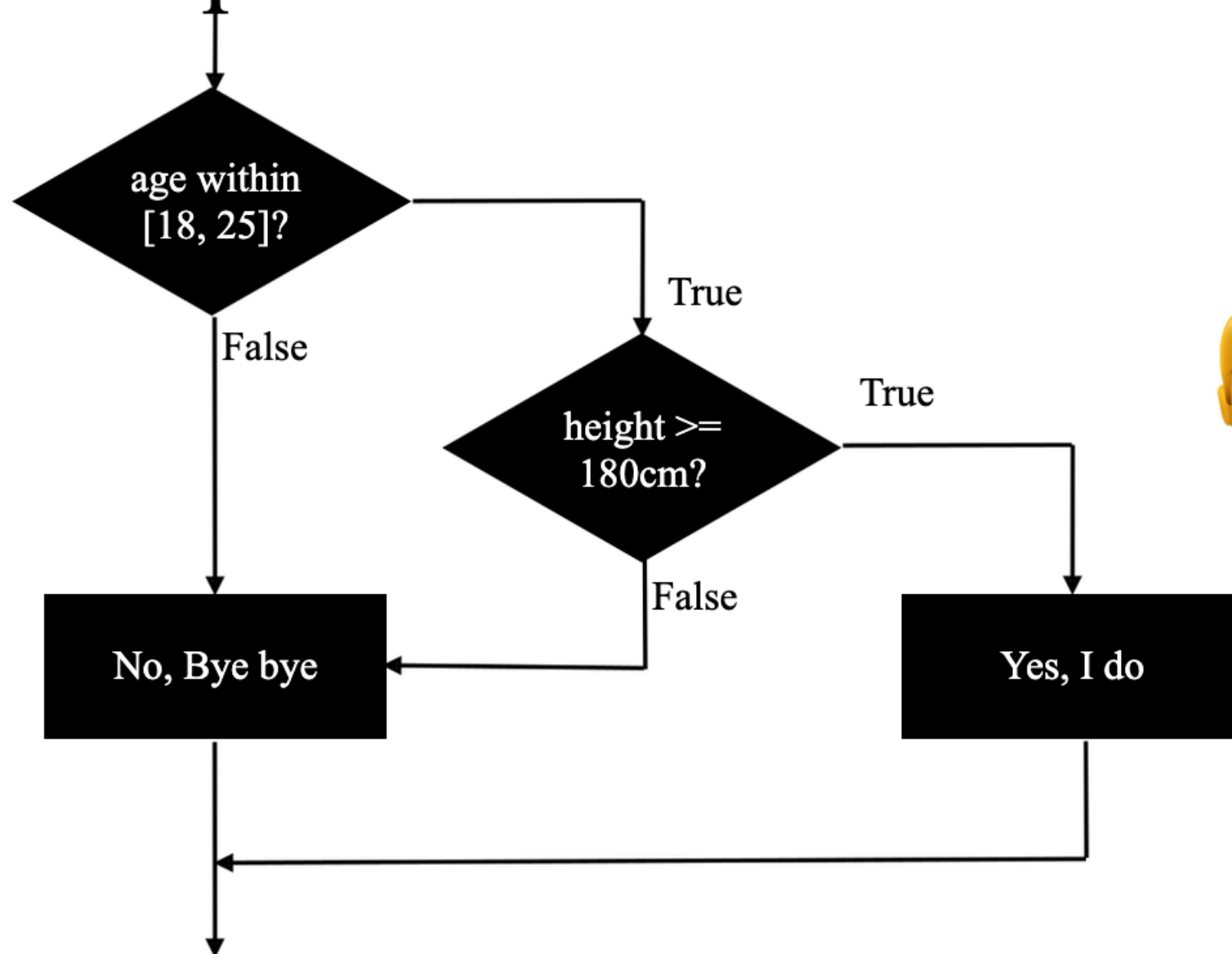
A Flowchart Shows the Program's Logic

```
PY main.py
1 age = 17
2 if (age >= 18):
3     print('You are an adult')
4 else:
5     print('You are under age')
```

Download: engg1330.dirk.hk/lec_02_example_57_24.py



Advanced Example



Expressions

Parentheses Bind Operators and Operands Together

PY main.py

```
1 x = 42
2 xMod2 = x % 2
3 if (xMod2 == 0):
4     print(x, 'is even')
5 else:
6     print(x, 'is odd')
```

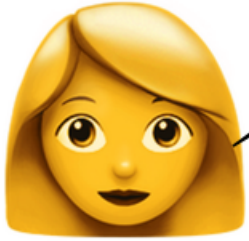
Download: engg1330.dirk.hk/lec_02_example_56_22.py

PY main.py

```
1 x = 42
2 if ((x % 2) == 0):
3     print(x, 'is even')
4 else:
5     print(x, 'is odd')
```

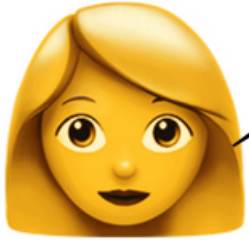
Download: engg1330.dirk.hk/lec_02_example_61_26.py

Expressions - Example



My Mr. Right must be 18 or older
and no older than 25 years old and
his height must be at least 180cm

Expressions - Example

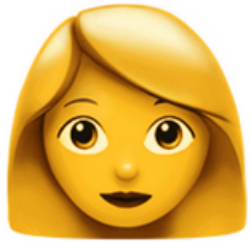
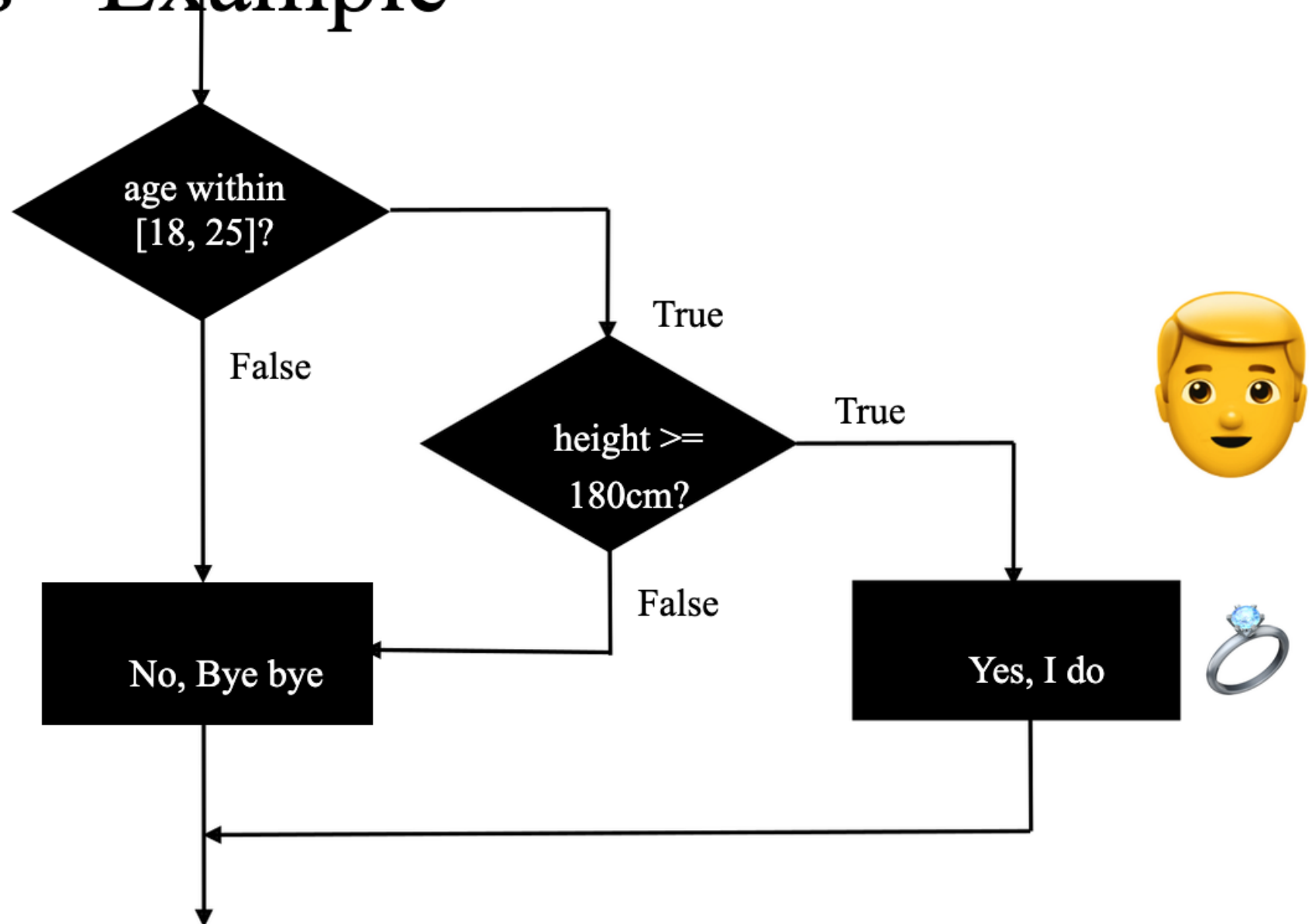


My Mr. Right must be 18 or older
and no older than 25 years old and
his height must be at least 180cm



Let's use a flow chart
to screen potential
boyfriends

Expressions - Example



Combining Conditions in an Expression

PY main.py

```
1 age = 25
2 height = 179
3 if ((age >= 18) and (age <= 25)) and height >= 180):
4     print('Yes, I do')
5 else:
6     print('No, Bye bye')
```

Download: engg1330.dirk.hk/lec_02_example_64_27.py

Combining Conditions in an Expression

PY main.py

```
1 age = 25
2 height = 179
3 if ((age >= 18) and (age <= 25)) and height >= 180):
4     print('Yes, I do')
5 else:
6     print('No, Bye bye')
```

Download: engg1330.dirk.hk/lec_02_example_64_27.py



Which Operation Happens First?



A screenshot of a Python IDE window titled 'main.py'. The code editor shows a single line of Python code: `print(2 + 3 * 4)`. The code is color-coded: `print` is red, `(` is blue, `2` is purple, `+` is blue, `3` is purple, `*` is blue, `4` is purple, and `)` is blue. A vertical line is positioned at the start of the code. At the bottom of the IDE window, there is a download link: [Download: engg1330.dirk.hk/lec_02_precedence_demo.py](http://engg1330.dirk.hk/lec_02_precedence_demo.py).

Highest to Lowest

`**`

`*` `/` `//` `%`

`+` `-`

Comparisons

`not`

`and`

`or`

Which Operation Happens First?



```
PY main.py
1 print(2 + 3 * 4)
```

Download: engg1330.dirk.hk/lec_02_precedence_demo.py

Highest to Lowest

**

* / // %

+ -

Comparisons

not

and

or

Which Operation Happens First?

```
PY main.py
1 print(2 + 3 * 4)
2 print((2 + 3) * 4)
```

Download: engg1330.dirk.hk/lec_02_precedence_demo.py

Highest to Lowest

**

* / // %

+ -

Comparisons

not

and

or

Which Operation Happens First?

```
PY main.py
1 print(2 + 3 * 4)
2 print((2 + 3) * 4)
```

Download: engg1330.dirk.hk/lec_02_precedence_demo.py

Highest to Lowest

**

* / // %

+ -

Comparisons

not

and

or

Which Operation Happens First?

```
PY main.py
1 print(2 + 3 * 4)
2 print((2 + 3) * 4)
```

Download: engg1330.dirk.hk/lec_02_precedence_demo.py

Highest to Lowest

**

* / // %

+ -

Comparisons

not

and

or

Parentheses make the intended grouping explicit

Casting and Square Bracket Operator

input() Returns a String

```
PY main.py
1 age = input('age: ')
2 if (age >= 18):
3     print('You are an adult')
4 else:
5     print('You are under age')
```

Download: engg1330.dirk.hk/lec_02_example_67_28.py

A String and an Integer Cannot Be Ordered

```
PY main.py
1 age = input('age: ')
2 if (age >= 18):
3     print('You are an adult')
4 else:
5     print('You are under age')
```

Download: engg1330.dirk.hk/lec_02_example_67_28.py

Converting Values to a Different Type

Conversion	Resulting Type
<code>int(x)</code>	Integer
<code>str(x)</code>	String
<code>float(x)</code>	Float

int() Converts the Input to an Integer

```
PY main.py
1 age = input('age: ')
2 print(type(age))
3 age = int(age)
4 print(type(age))
5 if (age >= 18):
6     print('You are an adult')
7 else:
8     print('You are under age')
```

Download: engg1330.dirk.hk/lec_02_example_70_29.py

Accessing String Characters by Index

Python uses zero-based indexing

Expression	Character
<code>s[0]</code>	First
<code>s[1]</code>	Second
<code>s[-1]</code>	Last
<code>s[-2]</code>	Second last

Keep the index within the string's bounds

Accessing Characters by Index

```
PY main.py
1 name = input('name: ')
2 index = int(input('index: '))
3 print(name[index])
```

Download: engg1330.dirk.hk/lec_02_example_72_30.py

Coding Challenge

To Earn Your Participation Mark

- Complete the full survey during the lecture
- Pass all tests in **at least 2 of the 3 challenges** by **23:59 HKT today**

Both required · 1 mark · No partial credit