PDST
Professional Development Service for Teachers
An tSeirbhís um Fhorbairt Ghairmiúil do Mhúinteoirí

An Roinn Oideachais agus Scileanna
Department of Education and Skills

National Workshop 1

LEAVING CERTIFICATE
COMPUTER SCIENCE

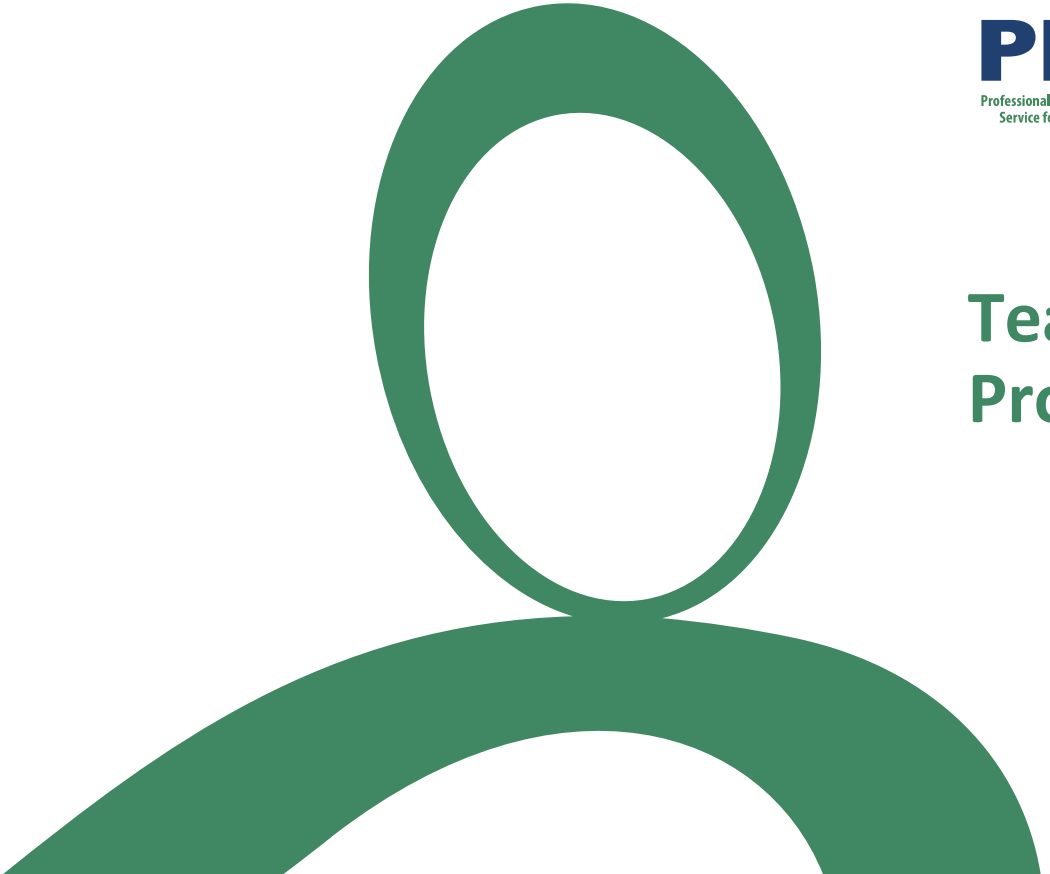
Question 1

What is the value of the variable `x` after the execution of this Python code?

python

```
x = 23
y = 17
x = x + y
x = y
```

A. 40
B. 57
C. 6
D. 17
E. None of the above



PDST
Professional Development Service for Teachers | An tSeirbhís um Fhorbairt Ghairmiúil do Mhúinteoirí

Teaching and Learning Programming

LCCS Strands

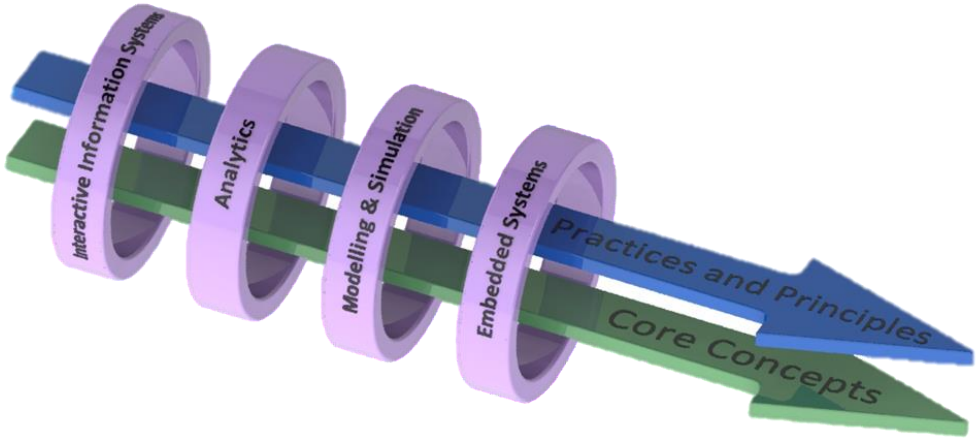
Strand 1: Practices and principles	Strand 2: Core concepts	Strand 3: Computer science in practice
▸ Computers and society ▸ Computational thinking ▸ Design and development	▸ Abstraction ▸ Algorithms ▸ Computer systems ▸ Data ▸ Evaluation/Testing	▸ Applied learning task 1 - Interactive information systems ▸ Applied learning task 2 - Analytics ▸ Applied learning task 3 - Modelling and simulation ▸ Applied learning task 4 - Embedded systems

Contents

- Introduction / Programming and LCCS
- Learning Challenges faced by Novice Programmers
- Phase 1 Teacher Input (related to teaching and related to students)
- Breakout #1 – MCQs
- Successful Strategies and Pedagogies used by Teachers
- Breakout #2 - PRIMM
- Conclusion (+ Resources and References)
- Q & A (via Slack)

Programming and LCCS

"The role of programming in computer science is like that of practical work in the other subjects—it provides motivation, and a context within which ideas are brought to life. Students learn programming by solving problems through computational thinking processes and through practical applications such as applied learning tasks." LCCS specification (2017)



The four applied learning tasks explore the four following contexts:

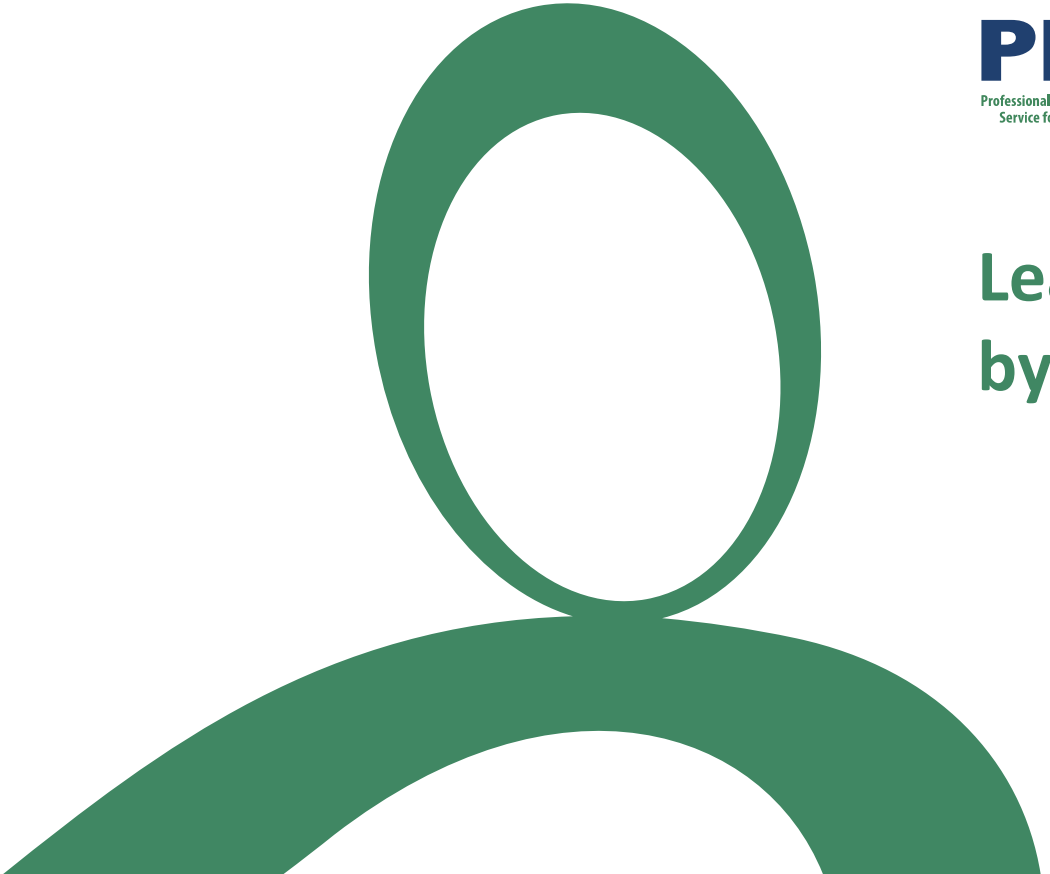
1 - Interactive information systems
2 - Analytics
3 - Modelling and simulation
4 - Embedded systems.

Question 2

What output does the Python code below display?

```
x = 0
y = (x == 21%7)
print(y)
```

A. 0
B. 3
C. False
D. True
E. None of the above



PDST
Professional Development Service for Teachers | An tSeirbhís um Fhorbairt Ghairmiúil do Mhúinteoirí

Learning Challenges faced by Novice Programmers

A Grand Challenge!

"Teaching and learning programming is considered one of the grand challenges of computing education"

Casperen, 2018

"Programming is a hard craft to master and its teaching is challenging"

Crick, 2017

"Programming is not an easy subject to be studied"

Lahtinen et. al., 2005

"Teaching students to program is a complex process"

Luxton-Reilly et. al., 2018

How did you learn how to program?

What were main challenges for you?

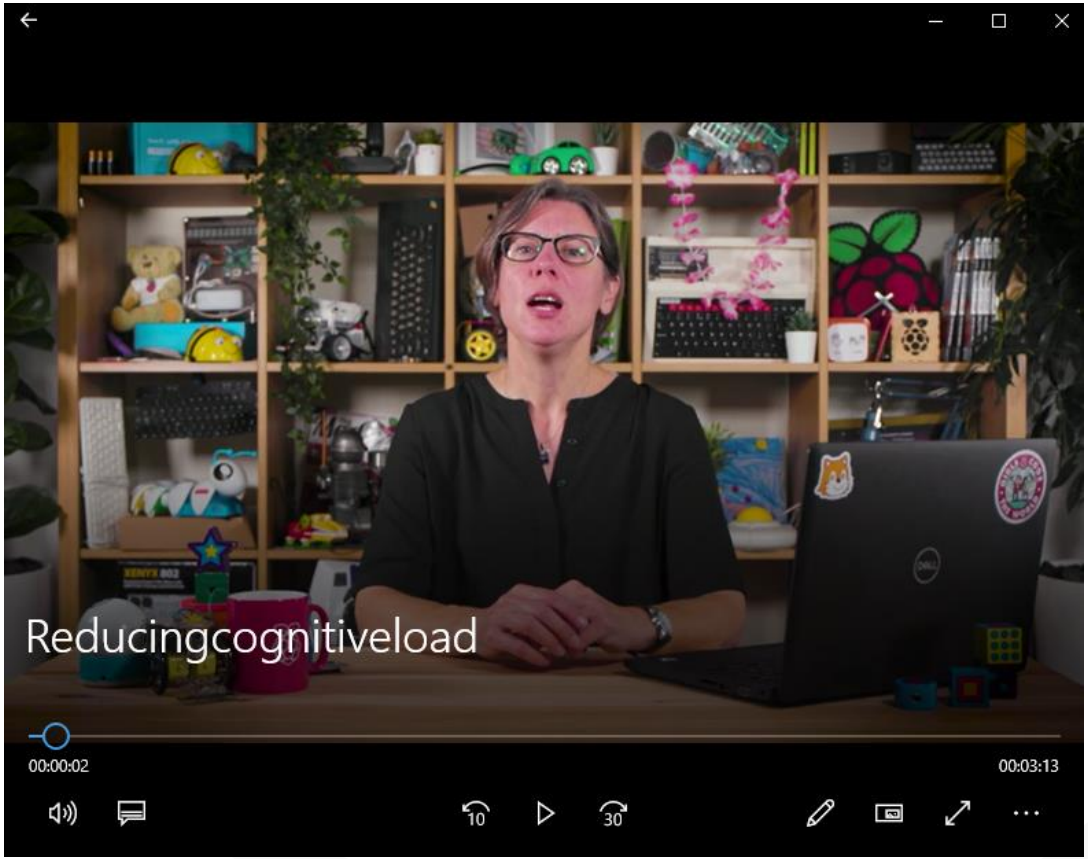
Learning Challenges faced by Novice Programmers

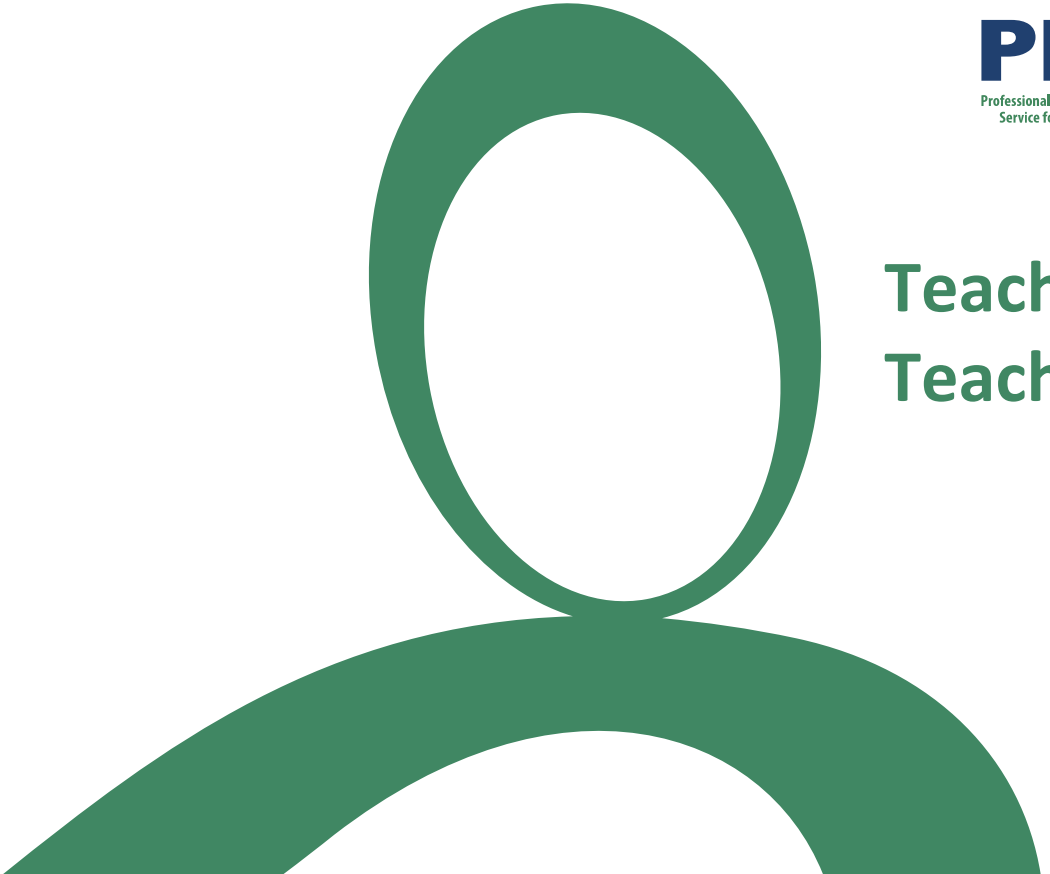
Programming IS difficult to learn because it requires and understanding of:

- the underlying machine that one is trying to control (Notational Machine)
- the problem to be solved (Orientation)
- the language syntax and semantics (Notation / Vocabulary)
- how to apply the language to solve problems (Application / Design)
- the development environment (Pragmatics and Perspectives)
- different levels of abstraction

Cognitive Load

Cognitive Load Video





PDST
Professional Development Service for Teachers | An tSeirbhís um Fhorbairt Ghairmiúil do Mhúinteoirí

Teaching Novice Programmers: Teacher Challenges

Phase 1 Teacher Input (Teacher Challenges)

Challenges relating to teachers

Challenges relating to students

Additional Challenges

- Digital Literacy
- Digital Divide
- Differentiation

- Differences from other subjects
- Perceptions and Expectations
- Misconceptions
- Teacher's Knowledge & Self-efficacy

Problem Solving Skills

- Students not engaged
- Students not practicing
- Students not understanding
- Time
- Attendance
- Homework

Source: Sentance, S., Csizmadia, A. Computing in the curriculum: Challenges and strategies from a teacher's perspective. *Educ Inf Technol* **22,** 469–495 (2017).

Question 3

Given x = 20 and y = 9 which of the following Python statements does NOT output the integer 2 exactly?

python

A. `print(x//y)`

B. `print(x%y)`

C. `print(int(x/y))`

D. `print(x/(y+1))`

E. `print(int(x/(y+1)))`

A.
B.
C.
D.
E.

5 minute stretch break

PDST
Professional Development Service for Teachers | An tSeirbhís um Fhorbairt Ghairmiúil do Mhúinteoirí

Breakout Activity #1

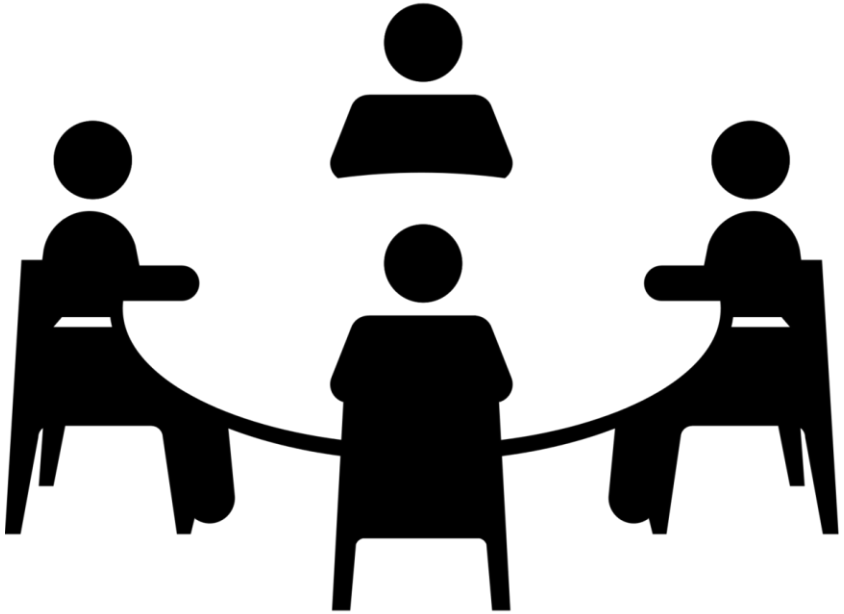
Question 4

What output does the Python code below display?

```
def calculate(y, x):
    a = x
    b = y + 1
    return a + b + y

x = 1
y = x + 1
print(calculate(x+1, y))
```

A. 4
B. 5
C. 6
D. 7
E. 8

Group Activity



PDST
Professional Development Service for Teachers | An tSeirbhís um Fhorbairt Ghairmiúil do Mhúinteoirí

Successful Strategies and Pedagogies for Teaching Programming to Novices

Peer Instruction

Well-evidenced pedagogical strategy

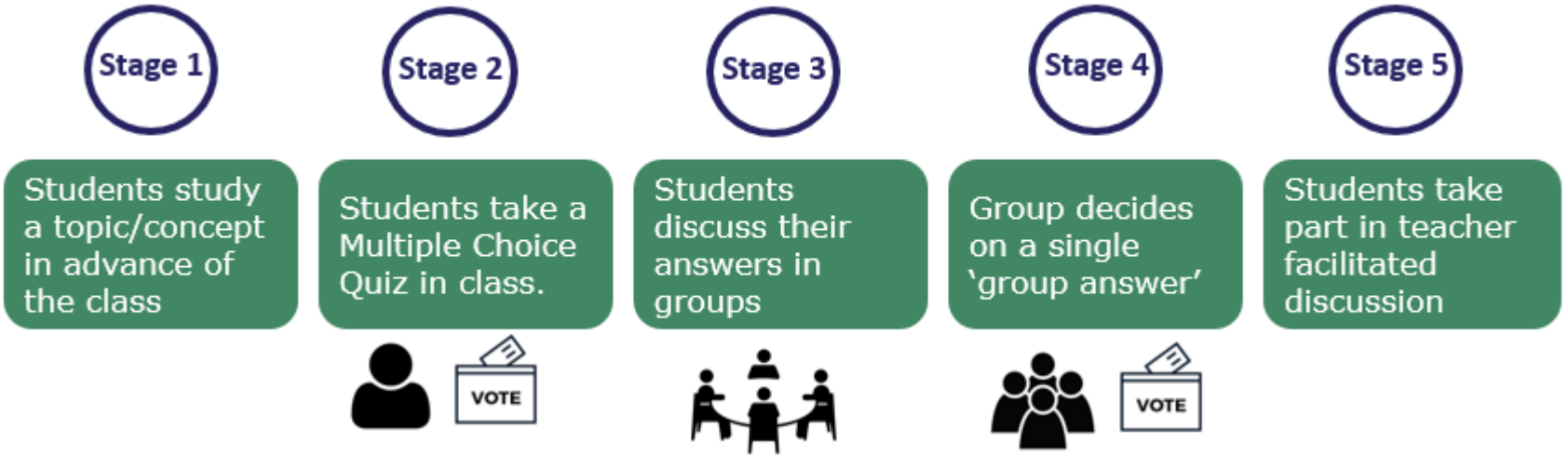
Combination of:

- Flipped learning
- Collaborative working
- Well-chosen MCQs

```
x = 0
y = (x == 21%7)
print(y)
```

A. 3
B. False
C. True
D. Syntax Error

Most effective where there are close distractors and known misconceptions



For more information on peer instruction see http://peerinstruction4cs.org

Successful Strategies and Pedagogies

Notational Machine

Topic Odering

Fix the syntax

Reflection

Problem Based Learning

Test Driven Development

Critical Reflection

Code Commenting

Program Tracing / Debugging

Find the 'bug'

Computational Discourse

PRIMM

(Use-Modify-Create)

Unplugged Activities

Peer Instruction

Game-based Pedagogy

Pair Programming

Block Programming

Inquiry Based Learning

Fill in the blanks

Turtle Graphics

Parson's Problems

Semantic Waves

Active Learning

Metacognition

Physical Computing

Modelling

Scaffolding

Progression

Context

Constructivism

Example: Fix the syntax

```
# Run the program to see what happens
# Can you fix the syntax error?
PRINT("Hello World")

# Now continue with the remaining 4 print statements ...
# You will need to uncomment each line and run the program to reveal each syntax error
#print(Hello World)
#print('Hello World")
#print "Hello World"
#print("Hello", World)
```

Follow-up Activity:

1. Question: Is the following (uncommented) line syntactically correct?
print("Hello", 123)

2. Create (and fix) your own syntax errors

3. Try to create an indentation syntax error … looks like this →

```
    print("Hello World")
    ^
IndentationError: unexpected indent
```

4. What happens if all the `print` statements are uncommented?

Example: Find the ‘bug’

```
# Find and fix the 'bug' in the program below

# The intention is to add a and b and display the answer

a = 3
b = 4
sum = a + 3
print(a, "+", b, "=", sum)
```

Follow-up Activity:

Would any of the following (uncommented) lines work in place of the print above
#print(3, "+", 4, "=", sum)
#print("3 + 4 =", sum)
#print("3 + 4 = 7")
#print(3 + 4)
#print(a + b)

Example: Insert comments

```
# Insert comments to explain each line of code below
# (the first one has been done to get you started)

x = 23 # Assign the value 23 to the variable x
y = 17
print("The value of x is", x)
print("The value of y is", y)
x = x + y
print("The value of x is", x)
x = y
print("The value of x is", x)
```

Example: Fill in the blanks

```
1.  # A program to demonstrate the multiple if statement
2.  import random
3.
4.  number = random.randint(1, 10)
5.  print(number) # comment this line out later!
6.
7.  guess = int(input("Enter a number between 1 and 10: "))
8.
9.  # Evaluate the condition
10. if guess == number:
11.       print("Correct")
12.       print("Well done!")
13. elif guess < number:
14.     print("Hard luck!")
15.     print("Too low")
16. else:
17.     print("Hard luck!")
18.     print("Too high")
19.
20. print("Goodbye")
```

```
if guess > number:
    [          ]
elif guess == number:
    [          ]
else:
    [          ]
```

Example 1: Parson's Problem

Arrange the blocks of code below into the correct order

```
elif guess < number:
    print("Hard luck!")
    print("Too low")
```

```
import random
number = random.randint(1, 10)
```

```
print("Goodbye")
```

```
else:
    print("Hard luck!")
    print("Too high")
```

```
if guess == number:
    print("Correct")
    print("Well done!")
```

```
guess = int(input("Enter a number between 1 and 10: "))
```

The final program should generates a random number, prompts the user to enter a guess and display a message telling the user if the guess was correct, too low or too high.

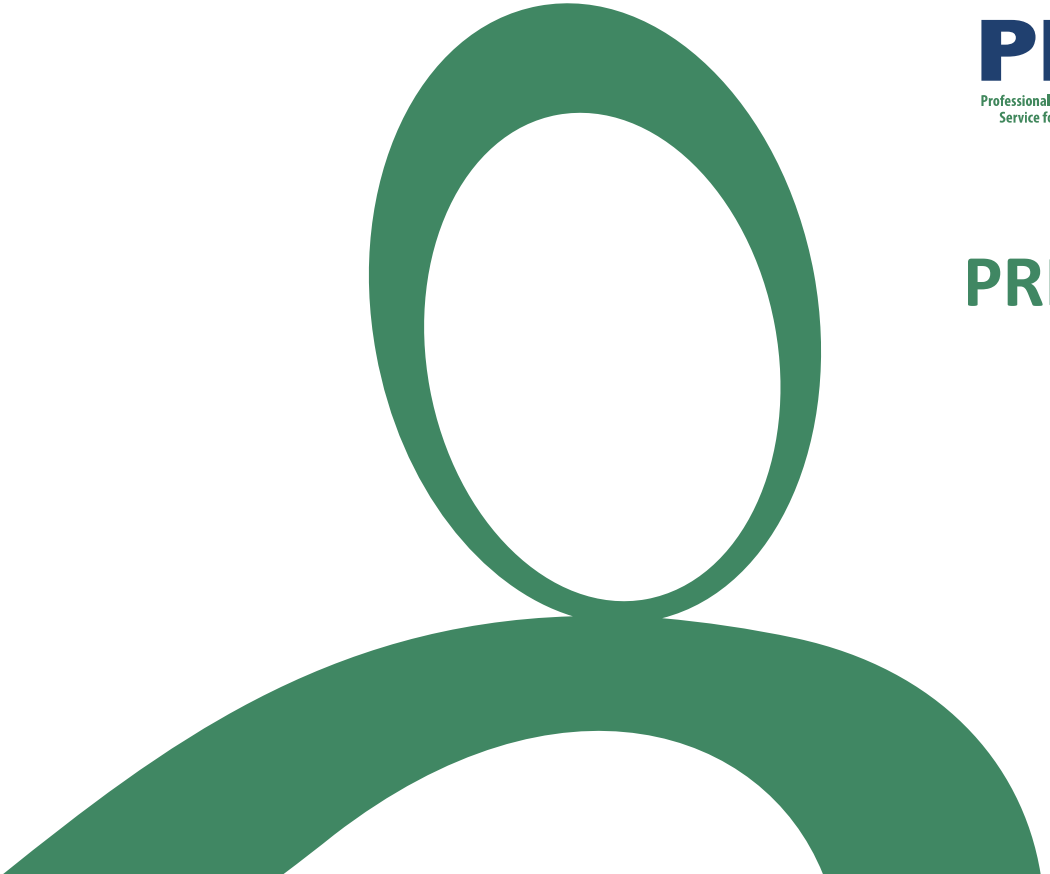
The program should always display the string *Goodbye* at the end.

Example 2: Parson's Problem

Re-arrange the jumbled up lines shown below so that the program prompts the end-user to enter two integers and then computes their and displays their sum.

```
number2 = int(number2)

number1 = int(input("Enter first number: "))

sum = sum + number1

number1 = int(number1)

print(number1, "+", number2, "=", sum)

number2 = input("Enter second number: ")

print("The answer is sum")

sum = number1 + number2
```

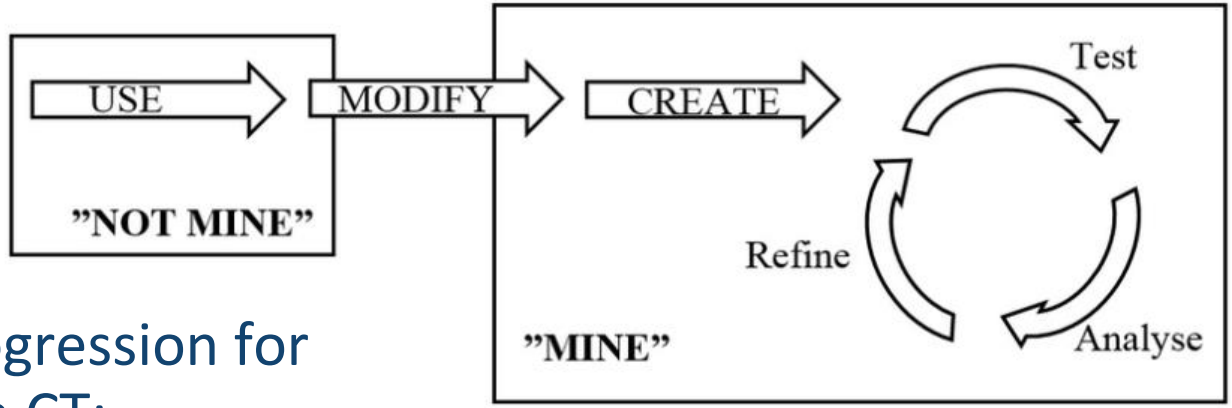
Warning! There are *three* extra lines that you won't need.



PDST
Professional Development Service for Teachers | An tSeirbhís um Fhorbairt Ghairmiúil do Mhúinteoirí

PRIMM

Use Modify Create



A three-stage progression for engaging youth in CT:

Stage 1: Students are consumers of someone else's creation

Stage 2: Students begin to modify the model, game or program

Stage 3: Through a series of modifications and iterative refinements …. What was someone else's becomes one's own

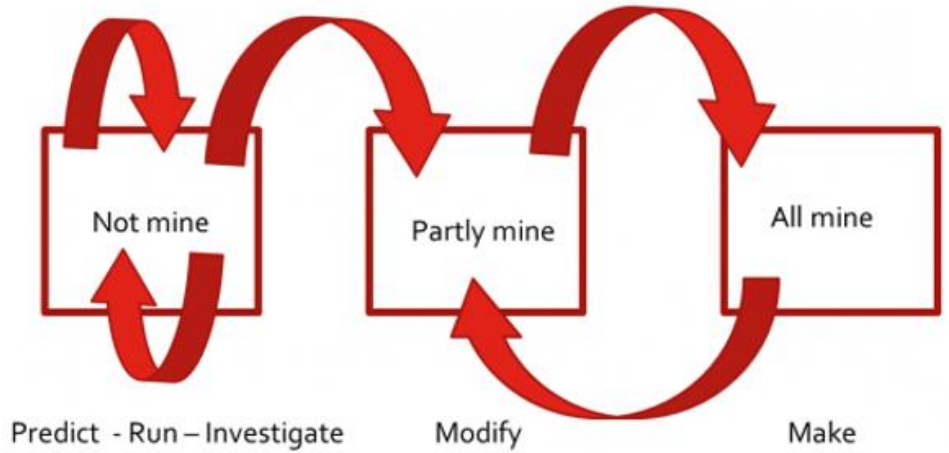
Tension between supporting student growth and limiting their anxiety. Teachers need to get the balance right.

Source: Lee, I., Martin, F., Denner, J., Coulter, B., Allan, W., Erickson, J., & Werner, L. (2011). Computational thinking for youth in practice. ACM Inroads, 2(1), 32–37.

PRIMM

A way of structuring programming lessons that focuses on:

- Reading before Writing
- Student Collaboration
- Reducing Cognitive Load
- Well-chosen starter programs
- Ownership Transfer



Sources:

1. https://blogs.kcl.ac.uk/cser/2017/02/20/exploring-pedagogies-for-teaching-programming-in-school/ (Sue Sentence)
2. https://blogs.kcl.ac.uk/cser/2017/09/01/primm-a-structured-approach-to-teaching-programming/ (Sue Sentence)
3. Sue Sentance, Jane Waite & Maria Kallia (2019) Teaching computer programming with PRIMM: a sociocultural perspective, Computer Science Education, 29:2-3, 136-176, DOI: 10.1080/08993408.2019.1608781

PRIMM

- **Predict:** given a working program, what do you think it will do? (at a high level of abstraction)
- **Run:** run it and test your prediction
- **Investigate:** What does each line of code mean? (get into the nitty gritty - low level of abstraction - trace/annotate/explain/talk about parts)
- **Modify:** edit the program to make it do different things (high and low levels of abstraction)
- **Make:** design a new program that uses the same nitty gritty but that solves a new problem.

PRIMM – Example (1 of 2)

```
1. import random
2.
3. number = random.randint(1, 10)
4. #print(number)
5.
6. guess = int(input("Enter a number between 1 and 10: "))
7.
8. if guess == number:
9.     print("Your guess was correct")
10.    print("Goodbye")
11.else:
12.    print("Incorrect guess")
13.    print("Goodbye")
```

Predict: Discuss in pairs. What do you think the above program will do? Be precise. Be succinct.

Run: Download the program / Key it in. Execute the program. Test your prediction. Were you correct?

Breakout Activity:

Investigate: Devise some questions to elicit student learning and curiosity. What if … Try … Explain …

Modify: Suggest some simple extensions / modifications for students to make in pairs. Same program.

Make: Formulate new problems that are conceptually similar. New context. New program (copy+paste)

PRIMM – Example (2 of 2)

```
1. import random
2.
3. number = random.randint(1, 10)
4. #print(number)
5.
6. guess = int(input("Enter a number between 1 and 10: "))
7.
8. if guess == number:
9.     print("Your guess was correct")
10.    print("Goodbye")
11.else:
12.    print("Incorrect guess")
13.    print("Goodbye")
```

Investigate:

1. Uncomment line 4. What happens?
2. What is the purpose of line 4?
3. What would happen if you removed `int` from line 6?
4. Try changing `==` to `!=` on line 8. What happens?
5. What if `==` was changed to `=` ?
6. What would happen if you don't enter an integer?
7. Try removing a bracket (anywhere). What happens?
8. Annotate each line of the program.

Modify:

1. Change the program so that it generates a number between 1 and 100? Can you be sure?
2. Change the program so that there is only one `print("Goodbye")` statement (without altering the logic)
3. Extend the program so that it tells the user if the number entered was *too high* or *too low*
4. Design an algorithm based on the program that would give the user 3 guesses
5. Get the computer to generate 4 numbers (lotto) OR ask the user how many numbers to generate?

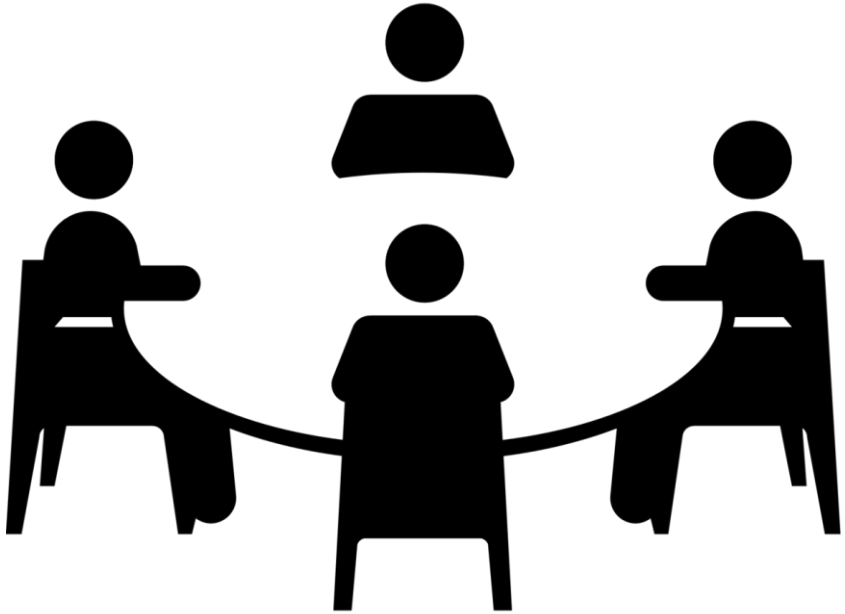
Make:

1. Write a program that generates two numbers and prompts the user to enter their product

PDST
Professional Development Service for Teachers | An tSeirbhís um Fhorbairt Ghairmiúil do Mhúinteoirí

Breakout Activity #2

PRIMM
Group Activity



PDST
Professional Development Service for Teachers | An tSeirbhís um Fhorbairt Ghairmiúil do Mhúinteoirí

Conclusions, Resources and References

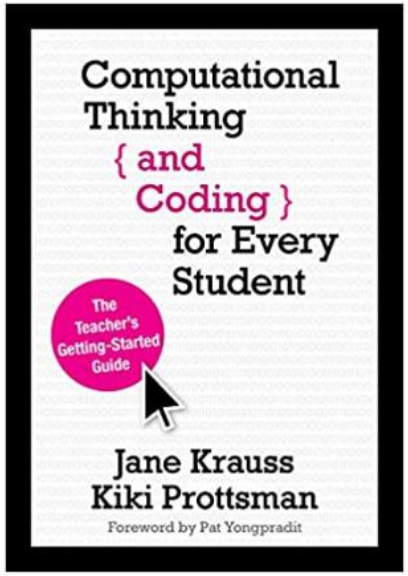
Conclusions

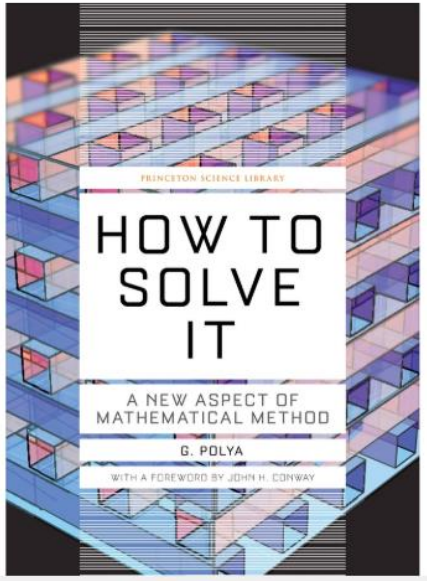
1. Programming IS difficult (for students to learn and teachers to teach)
2. Pedagogies are proven to work
3. Read/Trace code before you write
4. Constructivist approach is important
5. Growth mindset is *at least* as important as natural ability
6. Student-centric approach (teachers adopt a guide-on-the-side rather than a sage-on-the-stage approach)

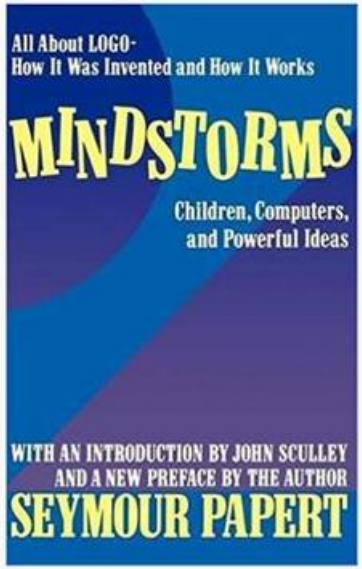
"The teacher should help, but not too much and not too little, so that the student shall have a reasonable share of the work" and, *"If the student is not able to do much, the teacher should leave him at least with some illusion of independent work."*

George Polya, How To Solve It

Resources







Plus lots more ….

COMPSCI.IE

Selection of References

Crick, Tom. *Computing Education: An Overview of Research in the Field*. Royal Society, 2017.

Lahiten et. al, (2005) A study of the Difficulties of Novice Programmers

Lee, I., Martin, F., Denner, J., Coulter, B., Allan, W., Erickson, J., & Werner, L. (2011). Computational thinking for youth in practice. ACM Inroads, 2(1), 32–37.

Luxton-Reilly, Andrew et. al. Introductory Programming: A Systematic Literature Review ITiCSE July 2018

Sentance, S., Csizmadia, A. Computing in the curriculum: Challenges and strategies from a teacher's perspective. *Educ Inf Technol* 22**,** 469–495 (2017).

Sue Sentance, Jane Waite & Maria Kallia (2019) Teaching computer programming with PRIMM: a sociocultural perspective, Computer Science Education, 29:2-3, 136-176

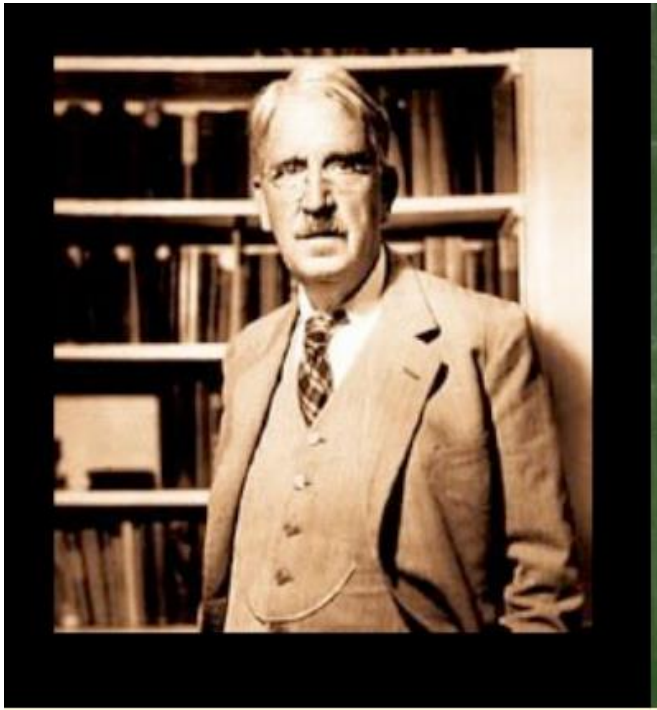
Robins et. al., (2003) Learning and Teaching Programming: A Review and Discussion

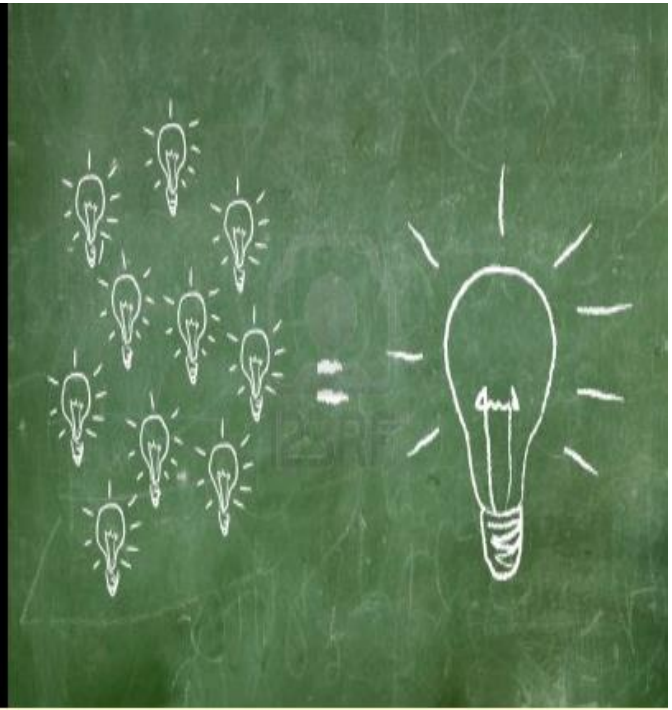
Yizhou Qian and James Lehman. 2017. Students' misconceptions and other difficulties in introductory programming: a literature review. ACM Trans. Comput. Educ. 18, 1, Article 1 (2017), 1:1–1:24 pages

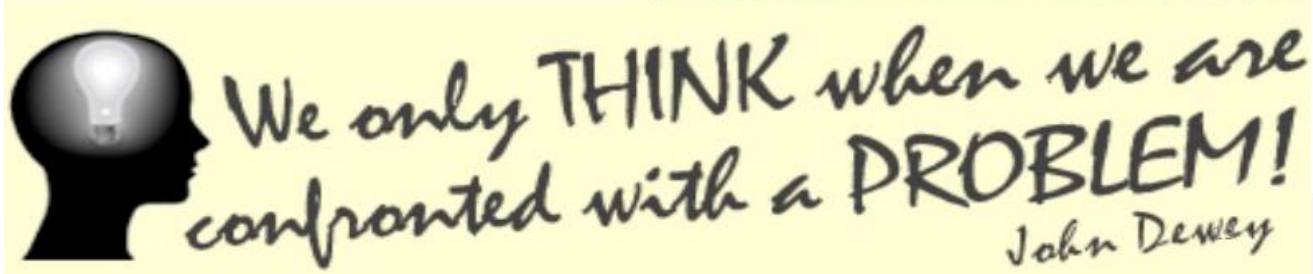
Waite, J. (2017) Pedagogy in teaching Computer Science in schools: A Literature Review. Royal Society.

Q & A

slack






We only THINK when we are confronted with a PROBLEM!
John Dewey

An Roinn Oideachais
agus Scileanna
Department of
Education and Skills