Activity #19: A Programming Language Tour
Recorder's Report

Manager:	Reader:
Recorder:	Driver:
Date:	Score: Satisfactory / Not Satisfactory

Record your team's answers to the key questions (marked with 🔑) below.

a) Model 1, Question #4

b) Model 2, Question #6

c) Model 3, Question #10

d) Model 4, Question #16

e) Model 5, Question #23

Activity #19: A Programming Language Tour

In this course, you will work in teams of 3–4 students to learn new concepts. This activity will introduce you to several different programming languages by having you implement the same algorithm in each language.

Content Learning Objectives

After completing this activity, students should be able to:

- Observe key differences between several programming languages

Process Skill Goals

During the activity, students should make progress toward:

- Write code to solve a problem in several different programming languages

Model 1 Computing the Value of π

$$\frac{\pi}{4} = \frac{1}{1} - \frac{1}{3} + \frac{1}{5} - \frac{1}{7} + \cdots$$

Refer to Model 1 above as your team develops consensus answers to the questions below.

Questions (10 min) Start time:

1. How does the **denominator** (bottom) change from term to term in this series of fractions?

2. How does the **sign** (plus or minus) change from term to term?

3. Fill in the missing values (white cells) in the table below to estimate π with 2, 3, 4, and 5 terms.

	$+\frac{1}{1}$	$-\frac{1}{3}$	$+\frac{1}{5}$	$-\frac{1}{7}$	$+\frac{1}{9}$	Approx. of $\frac{\pi}{4}$	Approx. of π
a. 1 term	+1.0					1.0	4.0
b. 2 terms	+1.0	−0.333					
c. 3 terms	+1.0	−0.333					
d. 4 terms	+1.0	−0.333					
e. 5 terms	+1.0	−0.333					

4. The infinite series above can be rewritten in several ways. Fill in the empty numerators in the series below to provide an equivalent series.

$$\pi = \frac{\quad}{1} + \frac{\quad}{3} + \frac{\quad}{5} + \frac{\quad}{7} + \frac{\quad}{9} + \cdots$$

Model 2 A Python Function to Estimate π

```
 1  # Function to estimate pi using a given number of terms
 2  def estimatePi(terms):
 3      if terms <= 0:
 4          return 0.0  # handle bad inputs
 5      numerator = -1.0
 6      total = 0.0
 7      for i in range(terms):  # loop from 0 to terms-1
 8          numerator *= -1
 9          denominator = 2 * i + 1
10          total += numerator / denominator
11      return 4 * total
12
13  # Test the estimatePi() function
14  def testEstimatePi():
15      vals = [-1, 0, 1, 2, 10, 100]
16      for n in vals:
17          print(n, estimatePi(n))
18
19
20  if __name__ == "__main__":
21      testEstimatePi()
22
```

Refer to Model 2 above as your team develops consensus answers to the questions below.

Questions (10 min) Start time:

5. The *Python* programming language was designed by Guido van Rossum to be clear and concise. Python has many specialized modules such as: *AstroPy* for astronomy, *PyGame* for video games, *SciPy* for scientific computing, and *PsychoPy* for behavioral science experiments. Answer the following questions based on the Python code above.

a) What character marks the start of a comment?

b) What character assigns a value to a variable?

c) What keyword starts an *if* statement?

d) What keyword starts a loop?

e) How are the values that the loop counter goes through given?

f) What keyword specifies the results of a function?

g) On what line is output printed?

h) What marks the start and end of a list of values?

i) What marks the start of the body of block of code (if, loop, function, etc)?

j) What marks the end of the body of block of code (if, loop, function, etc)?

6. How alike are Python and C++? Give at least two similarities and two differences.

7. An online Python interpreter is available at https://programiz.pro/ide/python. Copy the code in `activity19.py` into this interpreter and run it. What is the estimated value of π with 100 terms?

8. Now make the following changes to your program, testing those changes in the interpreter. Record how you implement each change.

a) Add 1000 to the list of test values.

b) Set `numerator = -4.0` on line 5 and adjust the code so that it still returns the correct value of π.

c) Change the loop starting on line 7 so that it steps through odd integers only (1, 3, 5, etc) and adjust the body of the loop so that the function still returns the correct value.

Model 3 A JavaScript Function to Estimate π

```
/* Function to estimate pi using the Leibniz formula */
function estimatePi (terms) {
  if (terms <= 0) {
    return 0.0
  }

  let numerator = -1.0
  let pi = 0.0

  for (let i = 0; i < terms; i++) { // loop from 0 to terms-1
    numerator *= -1
    const denominator = 2 * i + 1
    pi += numerator / denominator
  }

  return 4 * pi
}

/* Function to test the estimatePi() function */
function testEstimatePi () {
  const vals = [-1, 0, 1, 2, 10, 100]
  for (let i = 0; i < vals.length; i++) {
    console.log(vals[i] + ' ' + estimatePi(vals[i]))
  }
}

testEstimatePi()
```

Refer to Model 3 above as your team develops consensus answers to the questions below.

Questions (10 min)   Start time:

9. The *JavaScript* language runs in all major web browsers and is primarily used for client-side computation. Despite their names, JavaScript and Java are **not** closely related. Answer the following questions based on the JavaScript code above.

a) What character marks a comment?

b) What keyword specifies the results of a function?

c) What keyword declares a variable?

d) What command prints output?

e) What marks the start and end of a list of values?

f) What marks a block of code (body of if, loop, function, etc)?

10. How alike are JavaScript and C++? Give at least two similarities and two differences.

11. Give one way in which JavaScript is more similar to Python than it is to C++.

12. An online JavaScript interpreter is available at https://runjs.app/play. Copy the code in `activity19.js` into this interpreter and find the estimated value of π with 10 terms.

13. Now make the following changes to your program, testing those changes in the interpreter. Record how you implement each change.

a) Add 1000 to the list of test values.

b) Set `numerator = -4.0` on line 7 and adjust the code so that it still returns the correct value of π.

c) Change the loop starting on line 10 so that it steps through odd integers only (1, 3, 5, etc) and adjust the body of the loop so that the function still returns the correct value.

14. Indentation is required in Python, but not in C++ or JavaScript (or most other languages). Thus, the function `testEstimatePi()` could be written as:

```
function testEstimatePi() {var vals=[-1,0,1,2,10,100]; for (var
i=0;i<vals.length;i++){console.log(vals[i]+" "+estimatePi(vals[i]));}

```

Explain why it is still a good idea to indent code in these languages.

Model 4 An R Function to Estimate π

```r
# Function to estimate pi using a given number of terms
estimatePi <- function(terms) {
  if (terms <= 0) {
    return(0.0)  # check for bad inputs
  }

  numerator <- -1.0
  pi <- 0.0

  for (i in seq(0, terms - 1, 1)) {  # loop from 0 to terms-1
    numerator <- -numerator
    denominator <- 2 * i + 1
    pi <- pi + numerator / denominator
  }

  return(4 * pi)
}

# Function to test the estimatePi() function
testEstimatePi <- function() {
  vals <- c(-1, 0, 1, 2, 10, 100)
  for (i in vals) {
    cat(i, estimatePi(i), "\n")
  }
}

testEstimatePi()  # Run the test function

```

Refer to Model 4 above as your team develops consensus answers to the questions below.

Questions (10 min) Start time:

15. The *R* language was designed by Ihaka and Robert Gentlemen for statistical computing and graphics, based on the language *S*, originally developed by John Chambers at Bell Labs. Answer the following questions based on the R code above.

a) What character marks a comment?

b) What assigns a value to a variable (look for two options)?

c) What keyword starts an *if* statement?

d) What keyword starts a loop?

e) What keyword starts a function definition?

f) What keyword specifies the results of a function?

g) What command prints output?

h) What marks a block of code (body of if, loop, function, etc)?

16. Why might R's creators have chosen these unusual assignment operators? What confusion might it avoid?

17. Besides the assignment operator discussed above, give one difference between R code and each of Python, JavaScript, and C++ code.

18. An online R interpreter is available at https://www.programiz.com/r/online-compiler/. Copy the code in `activity19.R` into this interpreter and run it. What is the estimated value of π with 2 terms?

19. Now make the following changes to your program, testing those changes in the interpreter. Record how you implement each change.

a) Add 1000 to the list of test values.

b) Set `numerator = 4.0` on line 7 and adjust the code so that it still returns the correct value of π.

c) Change the loop starting on line 10 so that it steps through odd integers only (1, 3, 5, etc) and adjust the body of the loop so that the function still returns the correct value.

Model 5 Recursive Programs to Estimate π

A Recursive Python Program

```
1  # Function to estimate pi using recursion
2  def estimatePi(terms):
3      if terms <= 0:
4          return 0.0
5
6      val = 4.0 / (2 * terms - 1)
7      if terms % 2 == 0:
8          val *= -1
9
10     return val + estimatePi(terms - 1)
11
12 # Function to test estimatePi
13 def testEstimatePi():
14     vals = [-1, 0, 1, 2, 10, 100]
15     for i in vals:
16         print(i, estimatePi(i))
17
18 if __name__ == "__main__":
19     testEstimatePi()
20
```

A Scheme / LISP Program

```
1  ; Scheme function to estimate pi
2  (define estimatePi
3    (lambda (terms)
4      (if (<= terms 0)
5        0
6        (+ (estimatePi (- terms 1))
7          (/ (if (odd? terms) +4 -4)
8            (+ terms terms -1)
9            )))))
10
11 ; Function to test estimatePi
12 (define testEstimatePi
13   (lambda ()
14     (define vals `(-1 0 1 2 10 100))
15     (map estimatePi vals)
16   ))
17
18 (testEstimatePi)
19
```

Refer to Model 5 above as your team develops consensus answers to the questions below.

Questions (10 min) Start time:

20. The Python code above estimates π using the same series, but with a different approach. Answer the following questions based on the *Python* code above.

a) What variable(s) are defined in the `testEstimatePi()` function?

b) What function is called by `testEstimatePi()`?

c) What variable(s) are defined in the `estimatePi()` function?

d) What function is called by `estimatePi()`?

e) When `estimatePi(0)` is called, what value is returned?

f) When `estimatePi(1)` is called, what function call is made on line 10?

g) When `estimatePi(1)` is called, what value is returned?

h) When `estimatePi(2)` is called, what function call is made on line 10?

i) When `estimatePi(2)` is called, what value is returned?

21. A function that calls itself is called a *recursive* function. While possibly confusing at first, recursive functions can be both powerful and flexible. Which function(s) in the *Python* code above is recursive?

22. If recursive function *always* calls itself it will never end, and eventually the computer will run out of memory. Situations when a recursive function *does* call itself are called *recursive cases* and those when it *does not* are called *base cases*. What are the base and recursive cases for the function `estimatePi()`?

23. The *Scheme* language is a dialect of the *LISP* (short for LISt Processing) language, which was developed in the 1970's by Guy Steele and Gerald Sussman for artificial intelligence. Ideally, in Scheme programs the value of a variable never changes, and functions always return something useful. Thus, Scheme is called a *functional language*. Scheme also uses *recursion* (functions that call themselves) instead of loops. Answer the following questions based on the *Scheme* code above.

a) What marks a comment in *Scheme*?

b) What keyword assigns a value to a name?

c) What keyword defines a function?

d) What expression subtracts 1 from `terms`?

e) What expression checks whether `terms` is odd?

f) If that expression evaluates to true, what value is returned?

g) If that expression evaluates to false, what value is returned?

h) When `estimatePi(2)` is called, what function call is made?

24. You'll notice that Scheme uses parentheses quite a bit. Some people jokingly claim that "LISP" stands for "Lost In Stupid Parentheses". How many pairs of parentheses are used in defining each of the functions `estimatePi` and `testEstimatePi`?

25. An online `Scheme` interpreter is available at https://www.jdoodle.com/execute-scheme-online. Copy the Scheme code from `activity19.scm` into this interpreter and run it. Then determine the approximation of π with 10000 terms by adding 10000 to the list of test values.