

Objectives

- Conditional statements
- Exam review

Feb 17, 2021

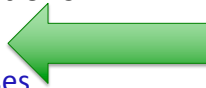
Sprenkle - CSCI111

1

1

Parts of an Algorithm

- Input, Output
- Primitive operations
 - What data you have, what you can do to the data
- Naming
 - Identify things we're using
- Sequence of operations
- Conditionals
 - Handle special cases
- Repetition/Loops
- Subroutines
 - Call, reuse similar techniques



Feb 17, 2021

Sprenkle - CSCI111

2

2

Making Decisions

- Sometimes, we do things only if some condition holds (i.e., “is true”)
- Examples
 - If the PB is new (has a safety seal)
 - Then, I will take off the safety seal
 - If it is raining and it is cold
 - Then, I will wear a raincoat
 - If it is Saturday or it is Sunday
 - Then, I will wake up at 9 a.m.
 - Otherwise, I wake up at 7 a.m.
 - If the shirt is purple or the shirt is on sale and blue
 - Then, I will buy the shirt

Feb 17, 2021

Sprenkle - CSCI111

3

3

Conditionals

- Sometimes, we only want to execute a statement in certain cases
 - Example: Finding the absolute value of a number
 - $|4| = 4$
 - $|-10| = 10$
 - To get the answer, we multiply the number by -1 *only if it's a negative number*
 - Code:

```
if x < 0 :  
    abs = x*-1
```

Feb 17, 2021

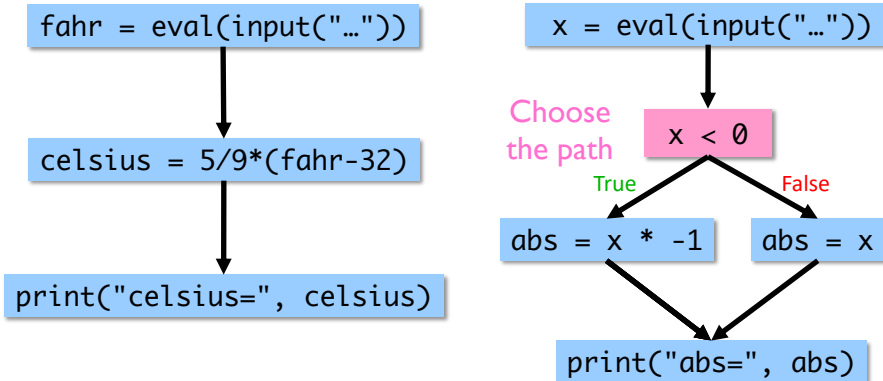
Sprenkle - CSCI111

4

4

if Statements

- Change the *control flow* of the program



Feb 17, 2021

Sprenkle - CSCI111

5

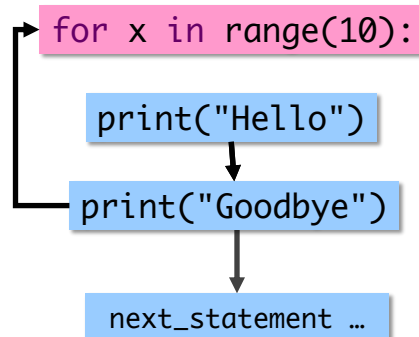
5

Other Constructs That Change Control Flow

- **for** loops

- Repeats a loop body a fixed number of times before going to the next statement after the **for** loop

```
for x in range(10):  
    print("Hello")  
    print("Goodbye")  
next_statement ...
```



Feb 17, 2021

Sprenkle - CSCI111

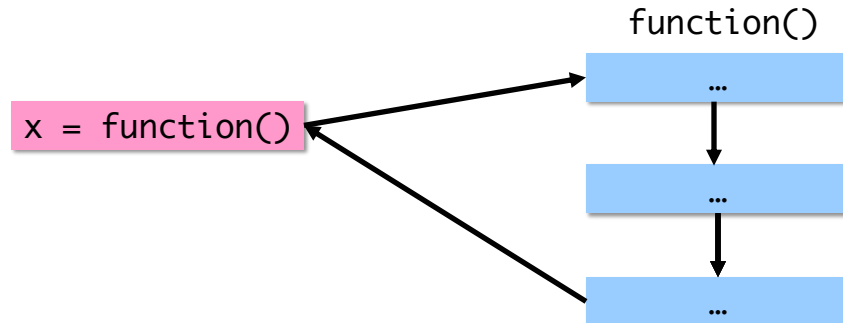
6

6

Other Constructs That Change Control Flow

- Function calls

- “Go execute some other code and then come back with the result”



Feb 17, 2021

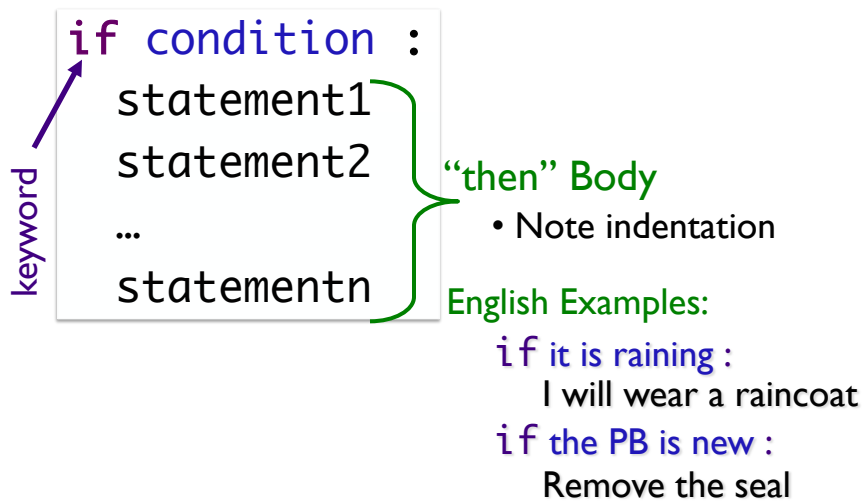
Sprenkle - CSCI111

7

7

Syntax of **if** statement:

Simple Decision



Feb 17, 2021

Sprenkle - CSCI111

8

8

Conditions

- Syntax (typical, others later):
 - `<expr> <relational_operator> <expr>`
- Evaluates to either `True` or `False`
 - Boolean type

Feb 17, 2021

Sprenkle - CSCI111

9

9

Relational Operators

- Syntax:
 - `<expr> <relational_operator> <expr>`
- Evaluates to either `True` or `False`
 - Boolean type

	Relational Operator	Meaning
Low precedence After arithmetic operators	<code><</code>	Less than?
	<code><=</code>	Less than or equal to?
	<code>></code>	Greater than?
	<code>>=</code>	Greater than or equal to?
	<code>==</code>	Equals?
	<code>!=</code>	Not equals?

Feb 17, 2021

Sprenkle - CSCI111

Use Python interpreter

10

10

Example: Using Conditionals

- Determine if a number is even or odd

```
x = eval(input("Enter a number: "))
remainder = x % 2
if remainder == 0 :
    print(x, "is even")
if remainder == 1:
    print(x, "is odd")
```

Feb 17, 2021

Sprenkle - CSCI111

evenorodd.py


11

11

Common Mistake: Assignment Operator vs. Equality Operator

- Assignment operator: =
- Equality operator: ==

```
x = eval(input("Enter a number: "))
remainder = x%2
if remainder = 0 :
    print(x, "is even.")
```



Feb 17, 2021

Sprenkle - CSCI111

12

12

Syntax of **if** statement:

Two-Way Decision

English Example:

if condition :
statement1
statement2
...
statementn } "then" Body

else :
statement1
statement2
...
statementn } "else" Body

keywords

if it is Saturday or it is Sunday :
I wake up at 9 a.m.
else :
I wake up at 7 a.m.

Feb 17, 2021

Sprenkle - CSCI111

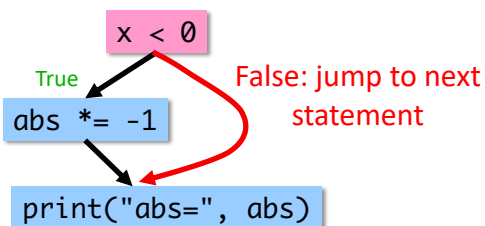
13

13

If-Else statements (absolute values)

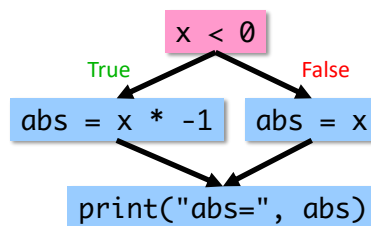
```
abs = x
if x < 0 :
    abs *= -1
print("abs=", abs)
```

If statement



```
if x < 0 :
    abs = x * -1
else:
    abs = x
print("abs=", abs)
```

If-else statement



Feb 17, 2021

Sprenkle - CSCI111

14

14

Example: Using Conditionals

- Determine if a number is even or odd
- More efficient implementation
 - Don't need to check if remainder is 1 because if it's not 0, it *must* be 1

```
x = eval(input("Enter a number: "))
remainder = x % 2
if remainder == 0:
    print(x, "is even")
else:
    print(x, "is odd")
```

Feb 17, 2021

Sprenkle - CSCI111

15

15

Practice: Draw the Flow Chart

```
print("This program determines your birth year")
print("given your age and current year")
print()
age = eval(input("Enter your age: "))

if age > 120:
    print("Don't be ridiculous, you can't be that old.")
else:
    currentYear = eval(input("Enter the current year: "))
    birthyear = currentYear - age
    print()
    print("You were either born in", birthyear, end='')
    print("or", birthyear-1)
print("Thank you. Come again.")
```

What does this code do?

Feb 17, 2021

Sprenkle - CSCI111

16

16

Function: max

- Given two numbers, returns the greater number

Feb 17, 2021

Sprenkle - CSCI111

17

17

Flow of Control: Using **return**

Is this implementation of the function correct?

Review: What does a return statement do/mean?

```
def max(num1, num2):  
    if num1 >= num2:  
        return num1  
    else:  
        return num2
```

Feb 17, 2021

Sprenkle - CSCI111

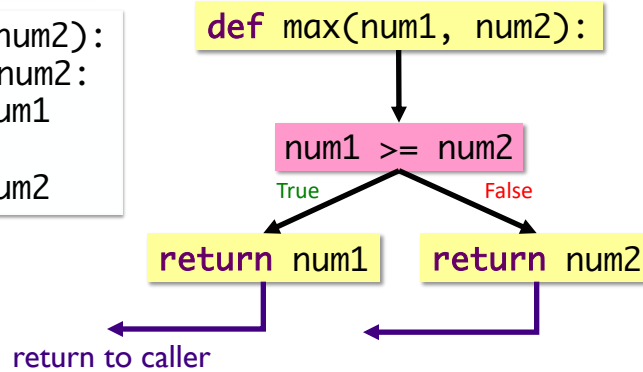
18

18

Flow of Control: Using **return**

Is this implementation of the function correct?

```
def max(num1, num2):  
    if num1 >= num2:  
        return num1  
    else:  
        return num2
```



Feb 17, 2021

Sprenkle - CSCI111

19

19

Flow of Control: Using **return**

Is this implementation of the function correct?

```
def max(num1, num2):  
    if num1 >= num2:  
        return num1  
    return num2
```

Feb 17, 2021

Sprenkle - CSCI111

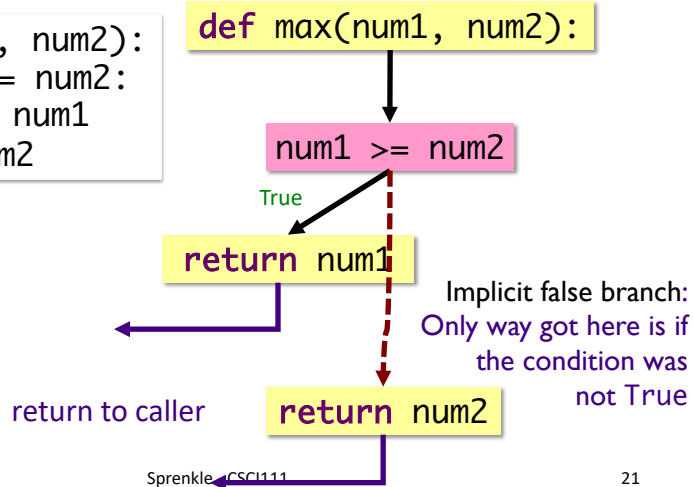
20

20

Flow of Control: Using `return`

Is this implementation of the function correct?

```
def max(num1, num2):  
    if num1 >= num2:  
        return num1  
    return num2
```



Feb 17, 2021

Sprenkle - CSCI111

21

21

Practice: Speeding Ticket Fines

- Any speed clocked over the limit results in a fine of at least \$50, plus \$5 for each mph over the limit, plus a penalty of \$200 for any speed over 90mph.
- Our function
 - Input: speed limit and the clocked speed
 - Output: the appropriate fine
 - What should the appropriate fine be if the user is not speeding?

`speedingticket.py`

Feb 17, 2021

Sprenkle - CSCI111

22

22

Test-Driven Development Process

1. Create test cases

- Known as test-driven development
- Idea: Focus on the outcomes first
- Helps you think about the problem without thinking about the code itself

2. Define function

Feb 17, 2021

Sprenkle - CSCI111

23

23

Example Test Cases

Speed limit	Clocked speed	Expected (fine)

Use these as tests in the test function:
`test.testEqual(calculateFine(speedLimit, clockedSpeed), fine)`

Feb 17, 2021

Sprenkle - CSCI111

24

24

Example Test Cases

Speed limit	Clocked speed	Expected (fine)
25	26	55
30	32	60
50	65	125
70	95	375
20	15	0
90	91	255
91	91	0

Use these as tests in the test function:
`test.testEqual(calculateFine(speedLimit, clockedSpeed), fine)`

Feb 17, 2021

Sprenkle - CSCI111

25

25

Looking Ahead

- Exam - Friday
 - Your Questions

Feb 17, 2021

Sprenkle - CSCI111

26

26