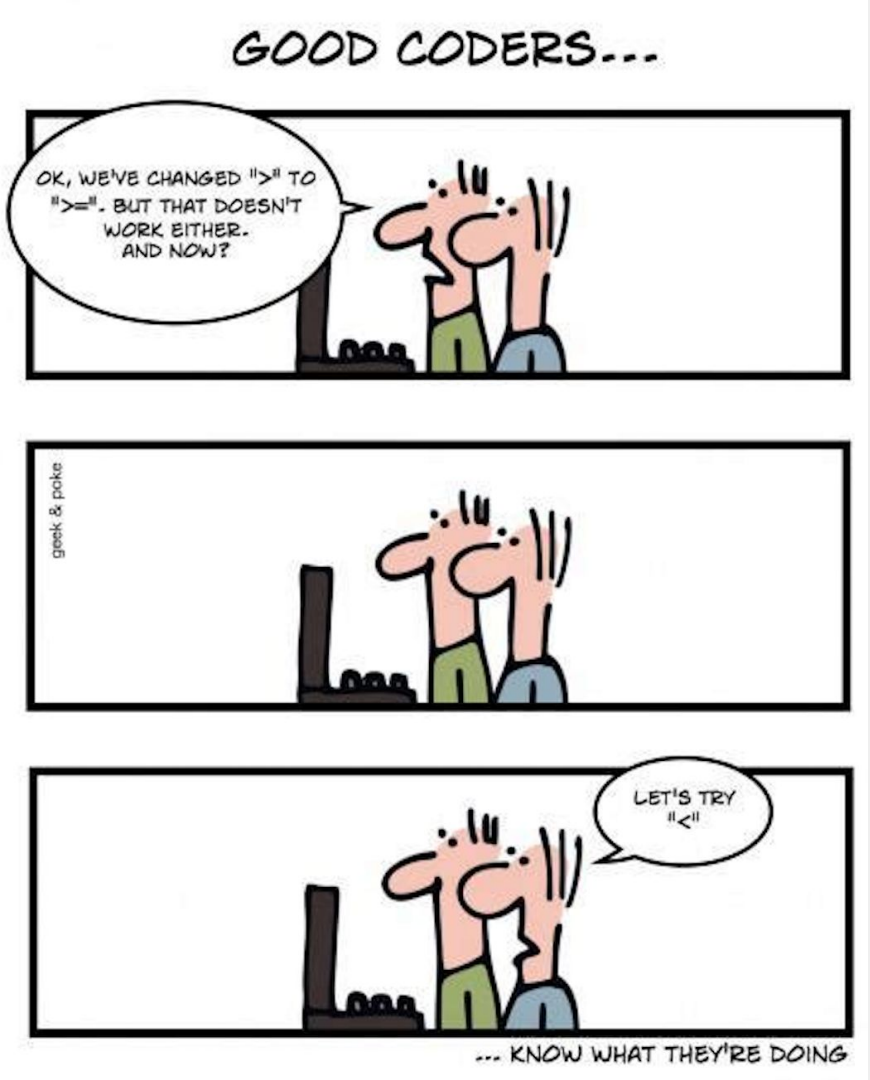


CS 110 while-loops

Adriana Picoral

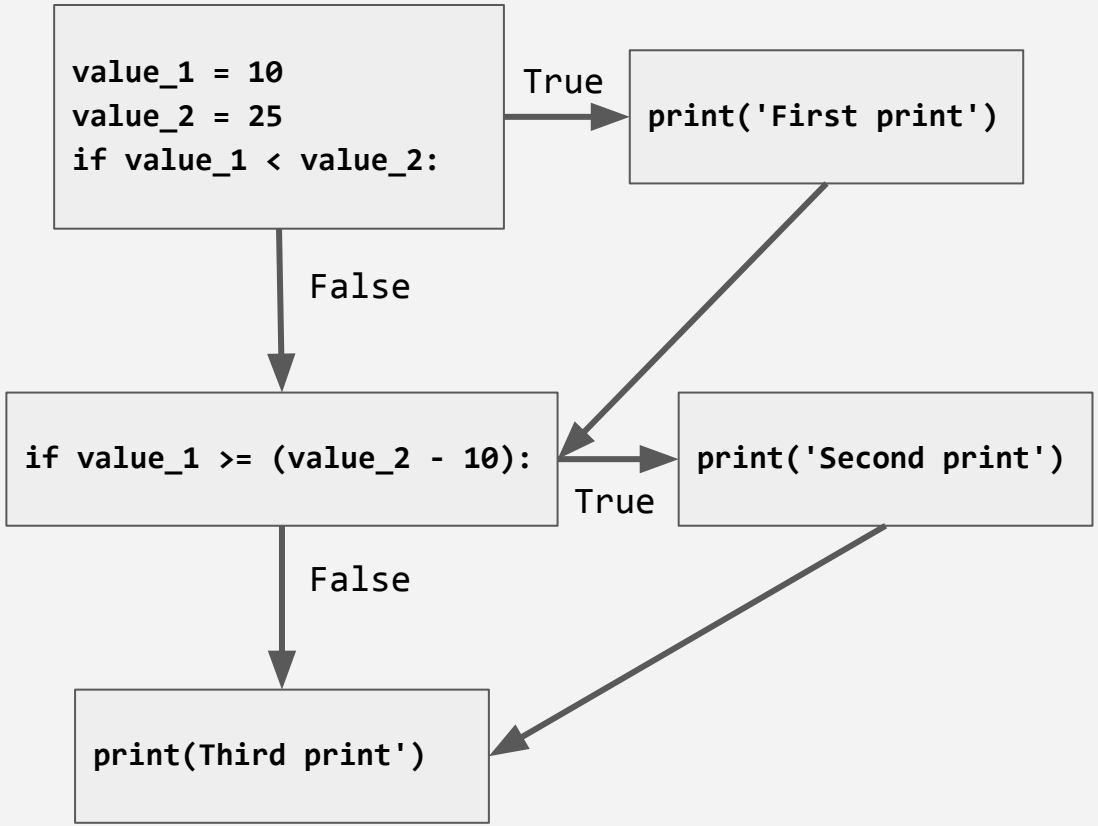
Other Announcements

- PA 4 Deadline
- Exam 1
  - Individual Exam 1, September 21
  - Group Exam 1, September 23
    - If more than 5 mins late to group, take on your own or with others who are late
  - Review Session: Tues, Sept 20, 5-7pm
  - Study Guide

```
weeks = int(input('How many weeks of work? '))
print('-------')

total_pay = 0
total_hours = 0

index = 1
while index <= weeks:
    pay = int(input('Week ' + str(index) + ' pay: '))
    hours = int(input('Week ' + str(index) + ' hours worked: '))
    total_pay += pay
    total_hours += hours
    index += 1

average_weekly_pay = total_pay / weeks
average_hourly_wage = total_pay / total_hours

print('-------')
print('Your average weekly pay was $' + format(average_weekly_pay, ',.2f'))
print('Your average hourly wage was $' + format(average_hourly_wage, ',.2f') + ' per hour')
```

Recall: Control-Flow Graph

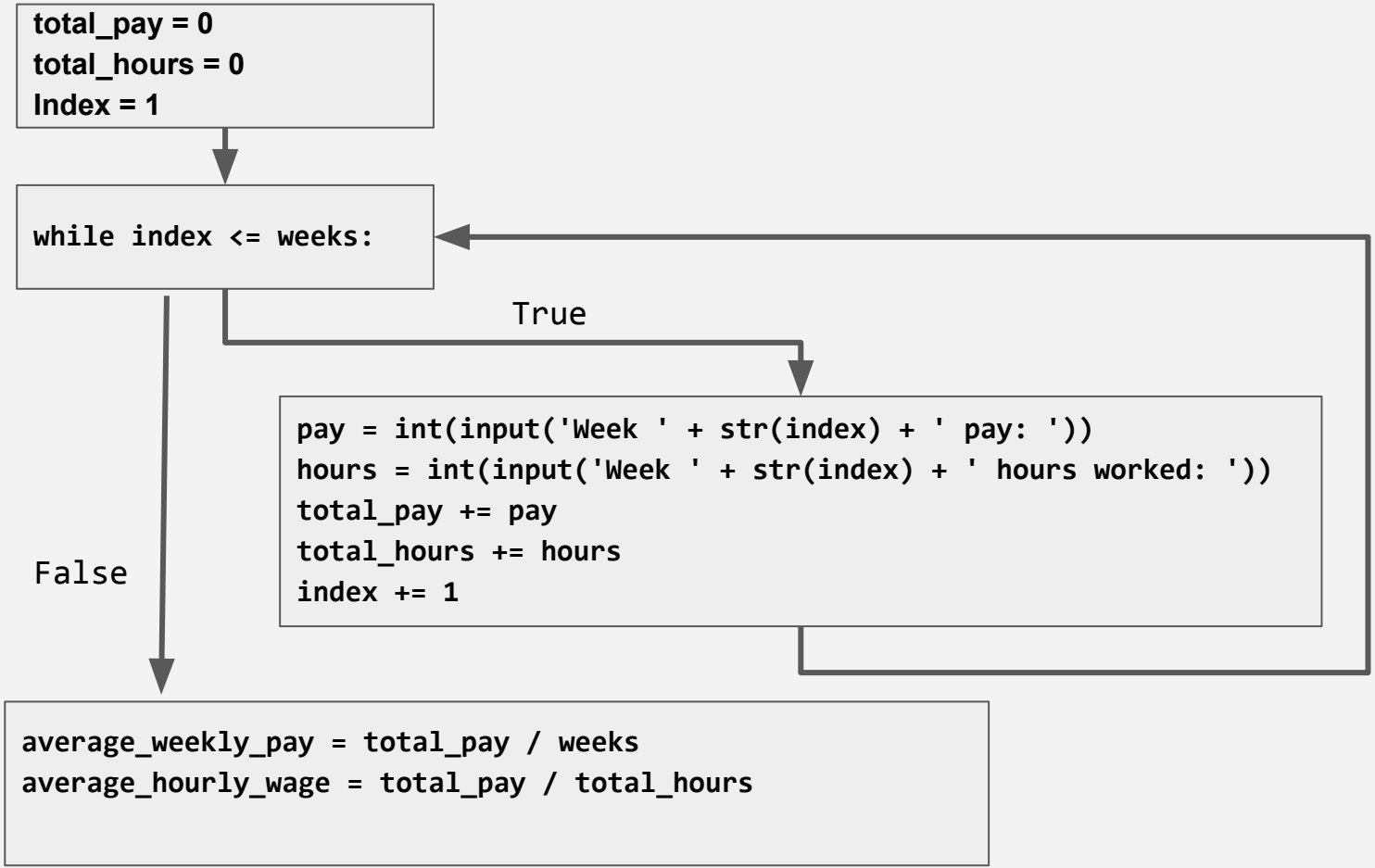
```
value_1 = 10
value_2 = 25

if value_1 < value_2:
    print('First print')

if value_1 >= (value_2 - 10):
    print('Second print')

print('Third print')
```



ICA

Draw the CFG

```
   . . .
 4 total_pay = 0
 5 total_hours = 0
 6
 7 index = 1
 8 while index <= weeks:
 9     pay = int(input('Week ' + str(index) + ' pay: '))
10     hours = int(input('Week ' + str(index) + ' hours worked: '))
11     total_pay += pay
12     total_hours += hours
13     index += 1
14
15 average_weekly_pay = total_pay / weeks
16 average_hourly_wage = total_pay / total_hours
   . . .
```


total_pay = 0
total_hours = 0
Index = 1
while index <= weeks:
True
False
pay = int(input('Week ' + str(index) + ' pay: '))
hours = int(input('Week ' + str(index) + ' hours worked: '))
total_pay += pay
total_hours += hours
index += 1
average_weekly_pay = total_pay / weeks
average_hourly_wage = total_pay / total_hours

Implement Steps A

```
Enter number of steps: 2

##
####

Enter number of steps: 4

##
####
######
########
```

```
Enter number of steps: 7

##
####
######
########
##########
############
##############
```

Solution for Steps A

```
steps = int(input('Enter number of steps: '))
print()
index = 1
while index <= steps:
    print('##' * index)
    index += 1
```

Implement Steps B

```
Enter number of steps: 2

  ##
####

Enter number of steps: 4

      ##
    ####
  ######
########
```

```
Enter number of steps: 7

          ##
        ####
      ######
    ########
  ##########
############
##############
```

Solution for Steps B

```
steps = int(input('Enter number of steps: '))
print()
index = 1
while index <= steps:
    space = '  ' * (steps - index)
    step_row = '##' * index
    print(space + step_row)
    index += 1
```

Implement Pyramid

```
Enter number of steps: 2

 ##
####

Enter number of steps: 4

   ##
  ####
 ######
########
```

```
Enter number of steps: 7

      ##
     ####
    ######
   ########
  ##########
 ############
##############
```

Solution for Pyramid

```
steps = int(input('Enter number of steps: '))
print()
index = 1
while index <= steps:
    space = ' ' * (steps - index)
    step_row = '##' * index
    print(space + step_row)
    index += 1
```

Implement Christmas Tree

```
Enter number of steps: 2

 %%
****

Enter number of steps: 4

   %%
  ****
 ######
%%%%%%%
```

```
Enter number of steps: 10

          %%
         ****
        ######
       %%%%%%%
      **********
     ##########
    %%%%%%%%%%
   **********
  ##########
 %%%%%%%%%%%%
```

```
steps = int(input("Enter number of steps: "))
index = 0
while index <= steps:
    space = " " * (steps - index)
    if index % 3 == 0:
        step_row = "##" * index
    elif index % 3 == 1:
        step_row = "%%" * index
    else:
        step_row = "**" * index
    index += 1
    print(space + step_row)
```

String indexes

- Each character in a string is located at a particular **index**
- The index is zero-based

String indexes

- Each character in a string is located at a particular **index**
- The index is zero-based

```
name = 'jeremiah'
```

String indexes

- Each character in a string is located at a particular **index**
- The index is zero-based

```
name = 'jeremiah'
```

index	0	1	2	3	4	5	6	7
character	j	e	r	e	m	i	a	h

Checking the value of a character

```
name = 'jeremiah'
print( name )
print( name[0] )

character_3 = name[3]
print( character_3 )
```

What will this print?

```
sentence = 'mailed list'
char_1 = sentence[7]
char_2 = sentence[1]
char_3 = sentence[10]
char_4 = sentence[4]
print(char_1 + char_2 + char_3 + char_3 + char_4)
```

What will this print?

```
sentence = 'mailed list'
length = len(sentence)
char_1 = sentence[length]
char_2 = sentence[1]
char_3 = sentence[3]
print(char_1 + char_2 + char_3 + char_3)
```

What will this print?

```
sentence = 'mailed list'
length = len(sentence) - 1
char_1 = sentence[length]
char_2 = sentence[1]
char_3 = sentence[3]
print(char_1 + char_2 + char_3 + char_3)
```

Add the code to print ‘cores’

```
sentence = 'computer science'
```

Add the code to print ‘cores’

```
sentence = 'computer science'
char_1 = sentence[0]
char_2 = sentence[1]
char_3 = sentence[7]
char_4 = sentence[12]
char_5 = sentence[9]
print(char_1 + char_2 + char_3 + char_4 + char_5)
```