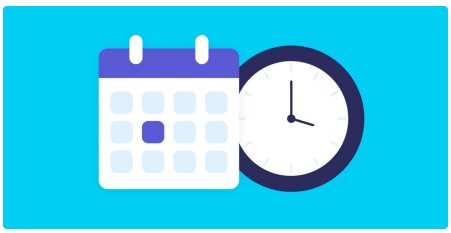
Python Programming for Beginners

Dr. Paul Bodily, bodipaul@isu.edu
https://www2.cose.isu.edu/~bodipaul

1

Schedule

- **Monday**: Think like a Programmer, Values, Data types, Variables, Input/Output
- **Tuesday**: For loops, Range Function, Turtle Graphics,
- **Wednesday**: Boolean values, If-Else Statements, Lists
- **Thursday**: Choose-Your-Own-Adventure
- **Friday**: Family Day



2

Resources

- **Slides and guided notes**: https://www2.cose.isu.edu/~bodipaul/outreach/pythonIntro/
- **E-book**: https://runestone.academy/ns/books/published/thinkcspy/index.html

Downloads:

- **PyCharm**: https://www.jetbrains.com/pycharm/download/
- **Python 3**: https://www.python.org/downloads/

3

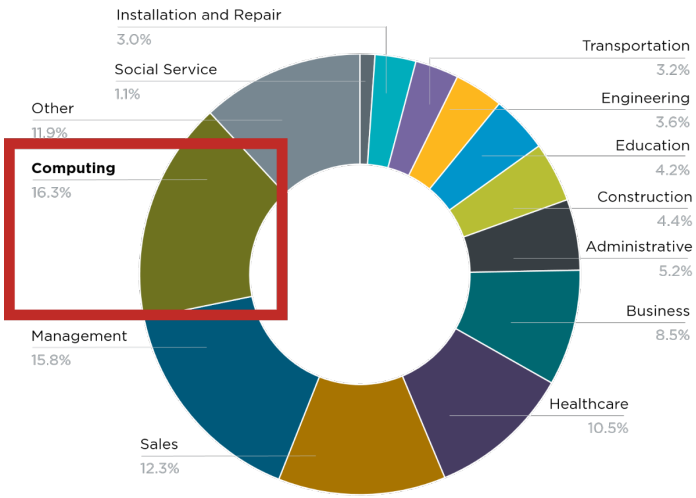
Top 10 Reasons to learn Computer Science

Pete Sanderson (Otterbein University) and Aisling Goodey (Perth College UHI)

1. _____
2. _____
3. _____
4. _____
5. _____
6. _____
7. _____
8. _____
9. _____
10. _____

4

There are lots of jobs: CS is #____ source of new wages in the US



500,000 current openings:
In **every** industry, **every** state, projected to grow at _____ the rate of all other jobs.

The picture in Idaho:
_____ open computing jobs
_____ CS graduates
_____ of high schools teach AP CS

Source: Code.org

5

High paying jobs, even as a student

$.58M
lifetime earnings of a high school graduate*

$1.19M
lifetime earnings of a college graduate*

*Net present value today

$1.67M
lifetime earnings of a computer science major*

Source: Brookings

Software Engineers in **Pocatello, ID:**

Average **Starting** Salary **:**
$_____

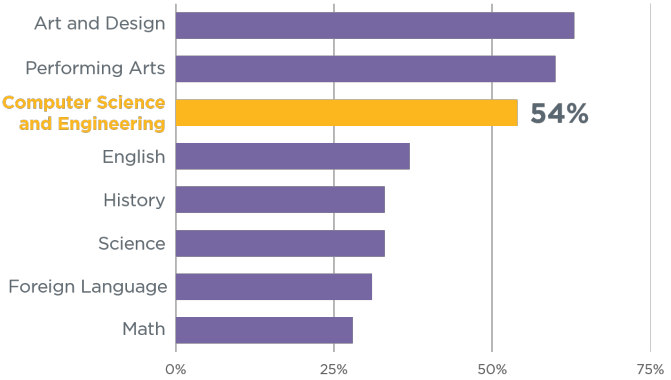
Source: www.roberthalf.com, May 2019

"You can earn money while you _____

6

High job satisfaction

Subjects students like the most:



Source:
Change the Equation

"For the **second year in a row**, software developer takes the _____ spot as the Best Job overall"

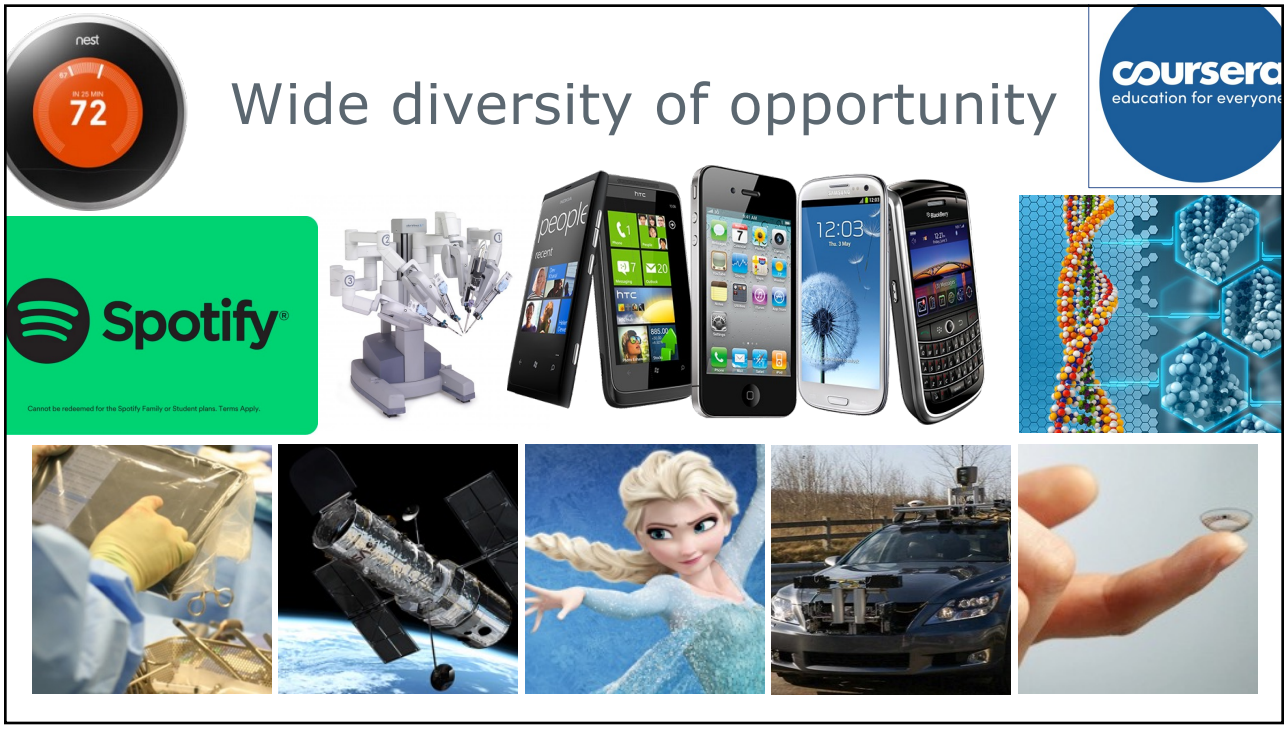
1. Software developer

Median salary: $101,790
Unemployment rate: 1.9 percent
Future job prospects: Good
Stress level: Average
Work-life balance: Above average

Source:
U.S. News (U.S. Bureau of Labor Statistics)

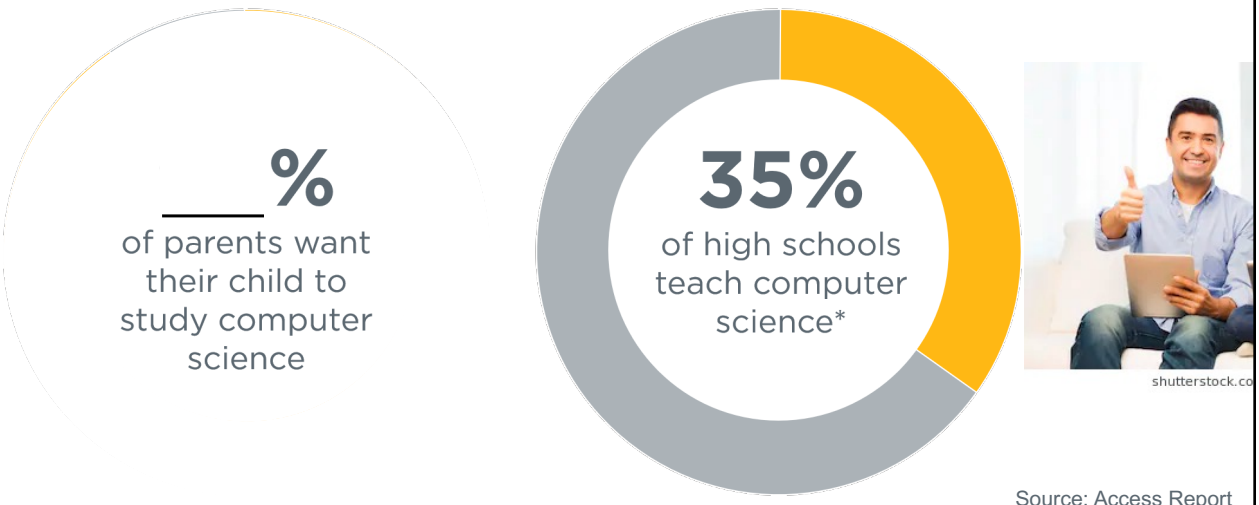
7

Wide diversity of opportunity



8

Studying CS makes your parents happy!

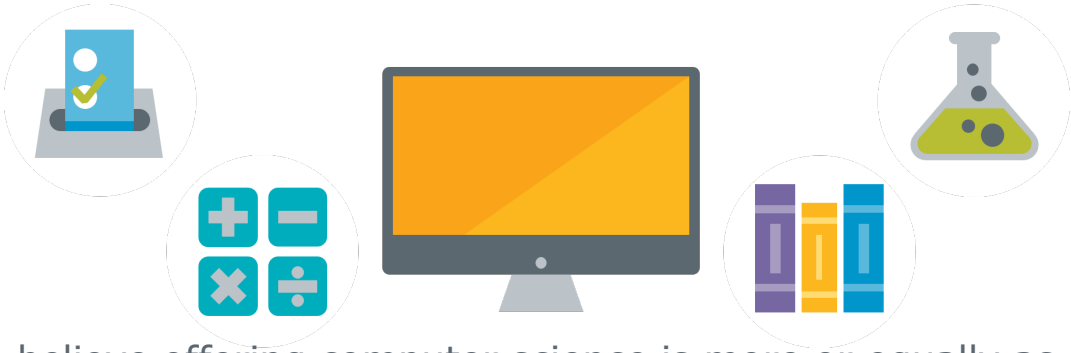


Source: Access Report Code.org

9

CS is an essential part of a well-rounded education

83% of parents and **64% of principals** in rural and small towns



believe offering computer science is more or equally as important as _____

Source: Code.org

10

11

12

E-book: https://runestone.academy/ns/books/published/thinkcspy/index.html

How to Think like a Computer Scientist Chapters

How to Think Like a Computer Scientist: Interactive Edition

About this Project

Table of Contents

15

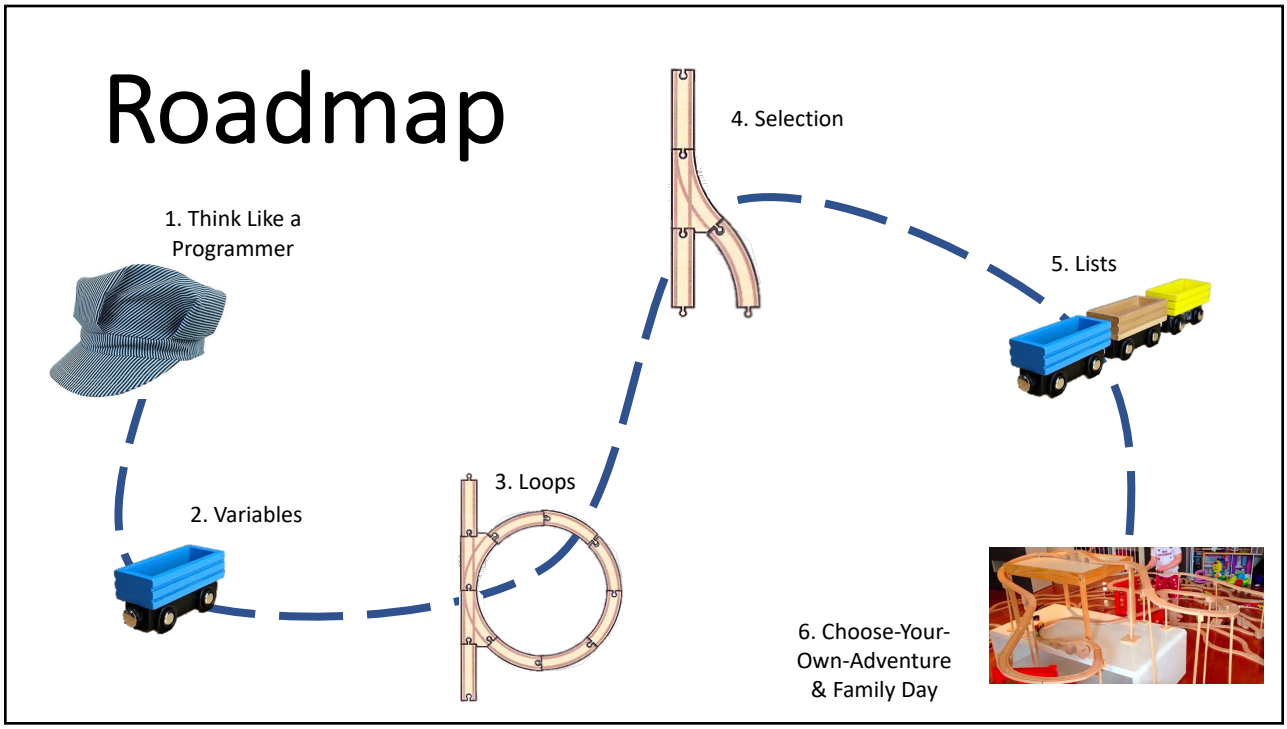
Python is a very popular language

"___ percent were satisfied or very satisfied with using Python,

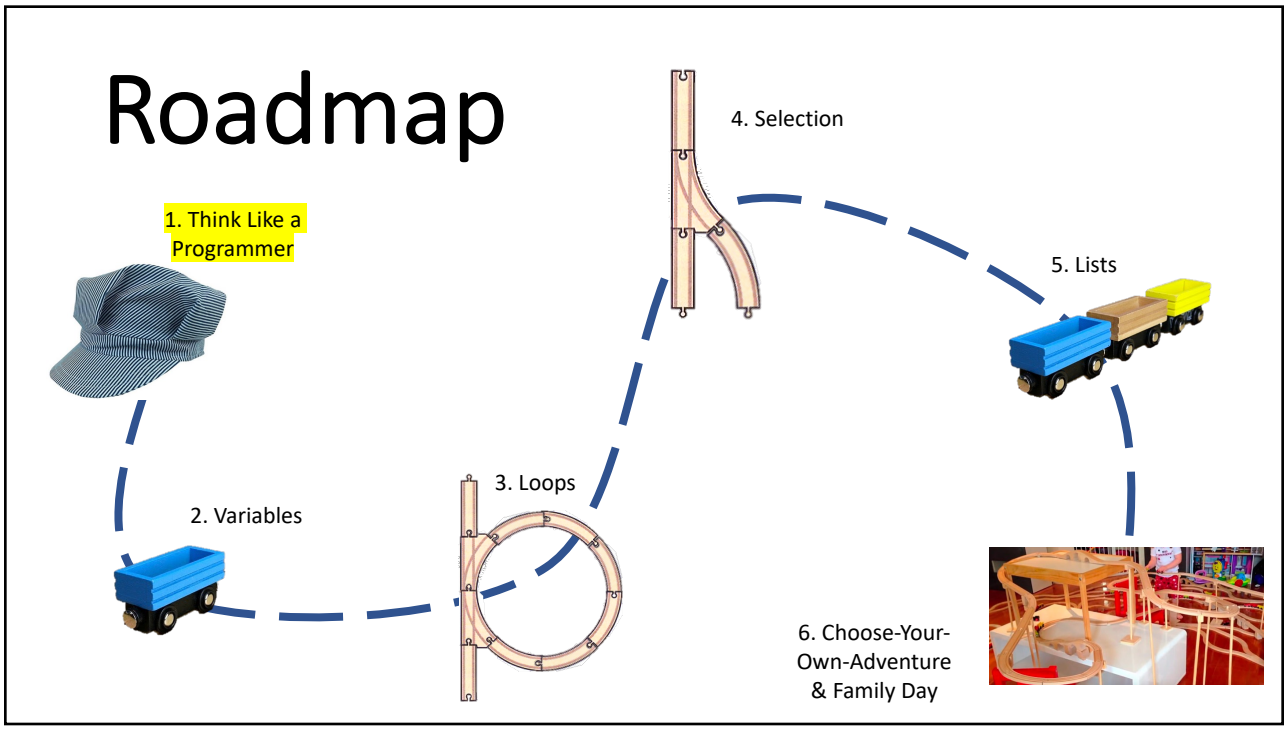
followed by C# ranked at a satisfaction level of ___ percent."

information-management.com

16


Roadmap
1. Think Like a Programmer
2. Variables
3. Loops
4. Selection
5. Lists
6. Choose-Your-Own-Adventure & Family Day

17


Roadmap
1. Think Like a Programmer
2. Variables
3. Loops
4. Selection
5. Lists
6. Choose-Your-Own-Adventure & Family Day

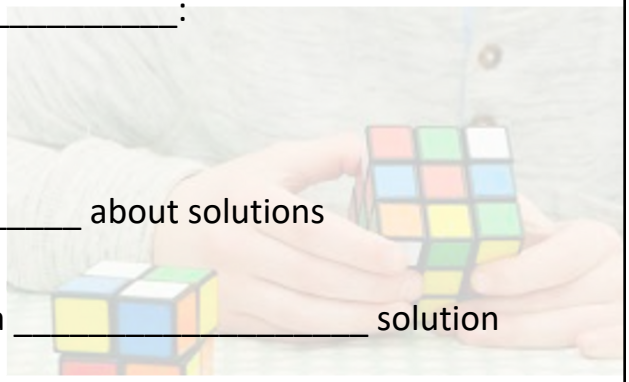
18

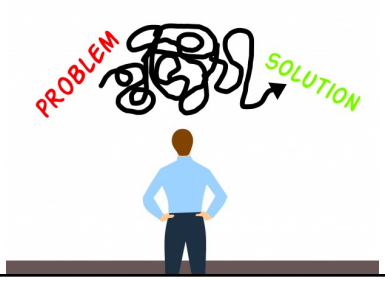
The single-most important skill for a computer scientist

Computer Science requires _____:

1. _____ problems

2. Think _____ about solutions





3. Express a _____ solution

19

Defining "algorithm" and "programming"

An **algorithm** is: a _____ that solves a problem

Programming takes an _____ and turns it into _____

20

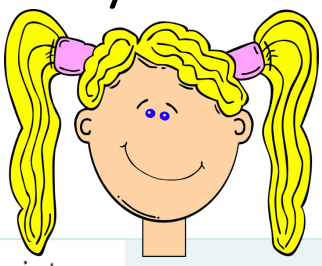
899
442
625
626
608
302
991
908
294
102
82
125
638
210
871
379
772
915
577
489
867
326
412
192
543

TAPPS—Describing an algorithm

- **Describe** an algorithm to find the **highest number** in a really long list of numbers. (Hint: do it yourself and then ask "how did I do it?")

21

Python is a *programming language*



Python is a

_____ and

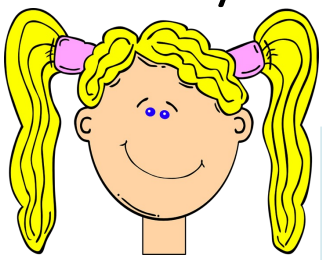
_____ language

script.py

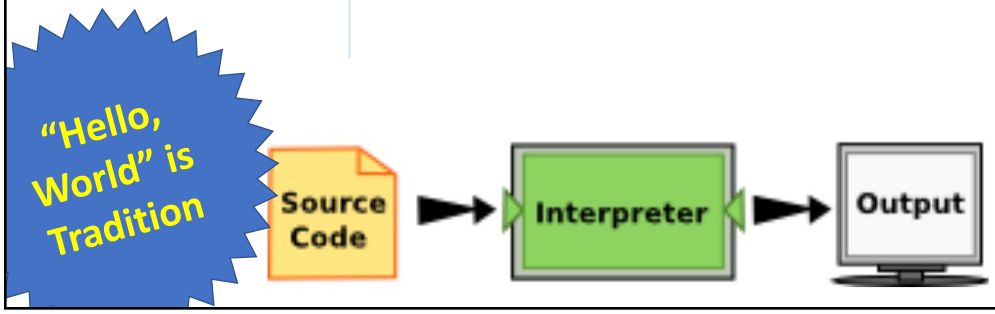
```
1  print("Ready...")
2  print("Get set...")
3  print("Go!")
```

22

You try it! Go to Phanon…



```
script.py
1   print("Hello, World!")
```



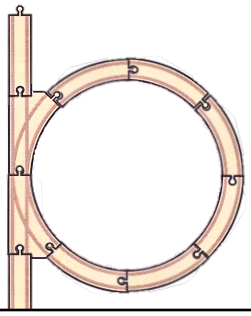
23

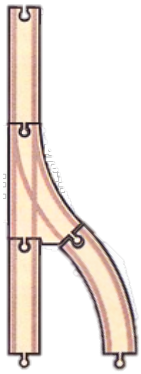
What all programs have in common

A **program** is a _____ of _____

***5* basic instructions** in every language:

1. _____
2. _____
3. _____
4. _____
5. _____





24

Programming errors are called “bugs”

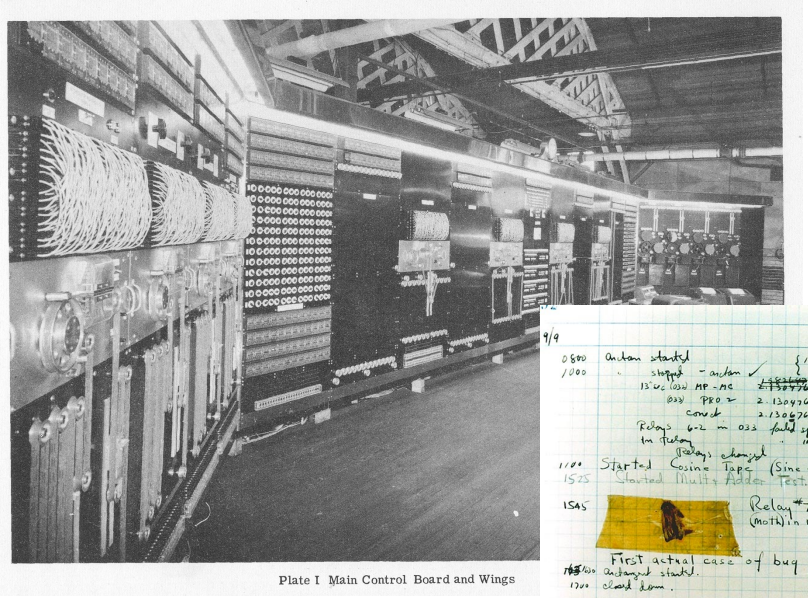
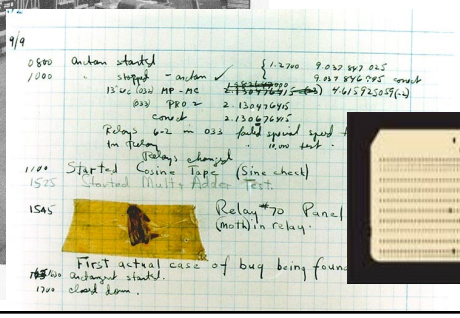
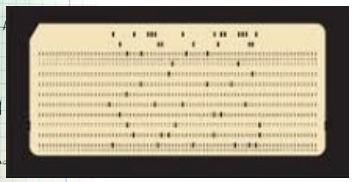


Plate I Main Control Board and Wings





_____ finds **syntax** and **runtime** errors

____ find **semantic** errors

25

How to detect and resolve ***Syntax*** errors

this sentence contains a syntax error. So does this one

- Syntax errors: _____ errors

script.py

```
1  print("Ready...")
2  prin(Set...")
3  print ( "Go!")
```

26

TAPPS—Find 3 Syntax Errors

- TAPPS—Find 3 things **wrong** with this code
- Try it in the eBook/Phanon/PyCharm
- Fix the bugs!

script.py

```
1  pint("Ready...")
2  print("Set...)
3  print Go!
```

27

How to detect and resolve ***Runtime*** errors

- Runtime errors are _____ detected by the_____
- Common runtime errors:
  - _____
  - _____

script.py

```
1  print("Ready...")
2  print("Set...")
3  print(3/0)
```

28

TAPPS—Experimenting with Runtime Errors

- Try this code in the eBook/Phanon/PyCharm
- Fix the code so that it reads
  `print(3/3)`

script.py

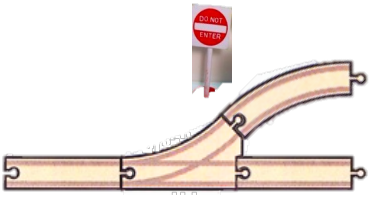
```
1  print("Ready...")
2  print("Set...")
3  print(3/0)
```

29

How to detect and resolve ***Semantic*** errors

Semantic errors_____errors detected by _____

"It's working, but it doesn't do what I want it to do"



script.py

```
1  print("Congratulations!")
2  print("Your percentage was:")
3  print(100/100)
```

30

TAPPS—Identify the type of error

- **Locate** 3 errors and **identify** the type of errors for each: syntax, runtime, or semantic

```
print(0/0,"people are in 1sth grade)
```

31

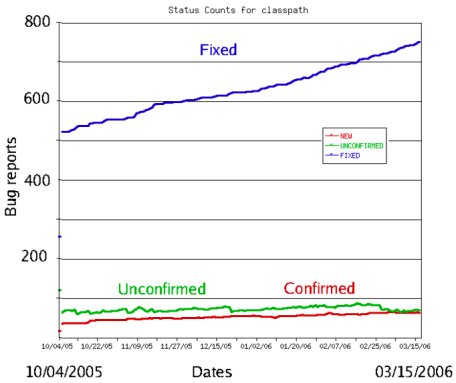
Debugging is a major part of programming

Debugging is like _____

Errors are your _____

IPython Shell

```
Traceback (most recent call last):
  File "<stdin>", line 3, in <module>
    print(3/0)
ZeroDivisionError: division by zero
```



Code a little, debug as you go, keep a working program

32

Good programmers comment their code

Comments lines start with ___

```
#-----------------------------------------------------
# This demo program shows off how elegant Python is!
# Written by Joe Soap, December 2010.
# Anyone may freely copy or modify this program.
#-----------------------------------------------------

# Greet the user
print("Hello, World!")

```

Use comments to _____

33

Real Python developers use an *IDE* to write Python code

IDE stands for _____
_____

It's like Microsoft Word for programmers!

*Increase the font size!

34

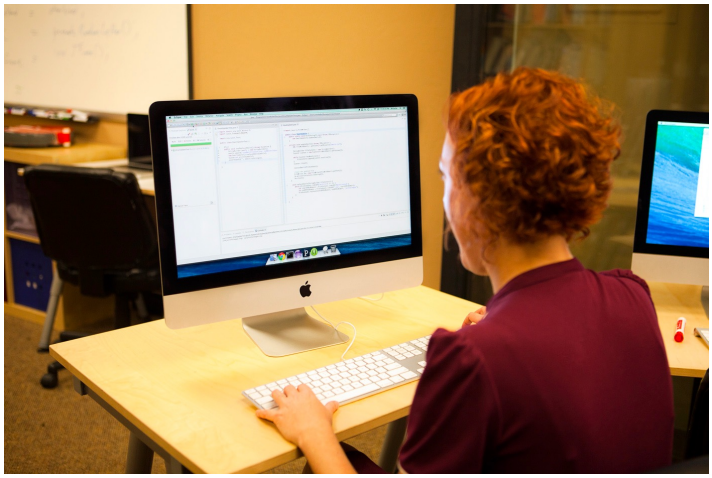
about_me.py: Write three Python statements that tell us something about you (use complete sentences)

- *Comment often and clearly*
- *Code a little, debug as you go, keep a working program*

Ask me to show you an example!

35

DISCUSS: Why is commenting important?



36

about_me.py: Write three Python statements that tell us what you like to do (use complete sentences)

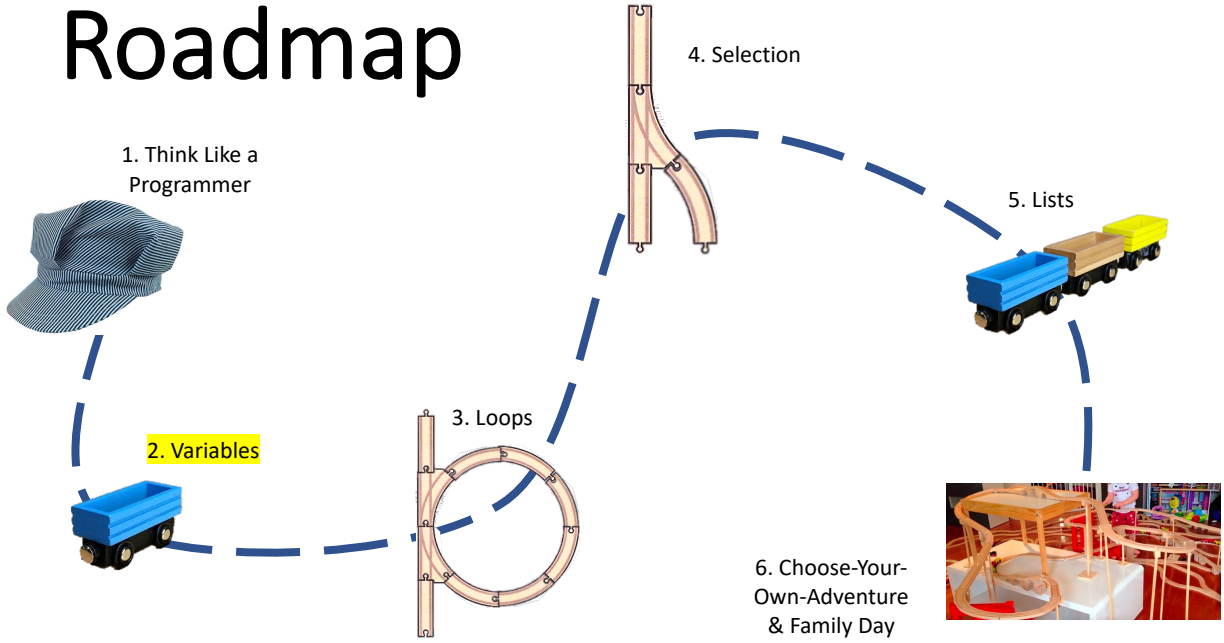
- *Comment often and clearly*
- *Code a little, debug as you go, keep a working program*

37

2-minute write: What have you learned so far about programming? Questions?

38

Roadmap

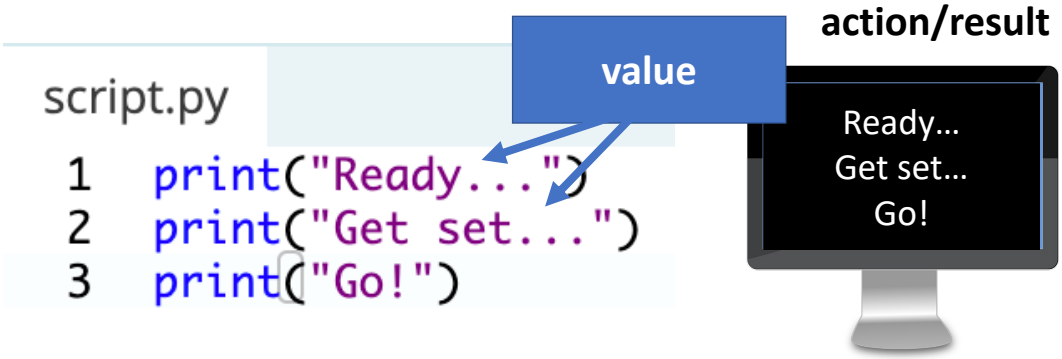


39

`print(…)` is a function

What does `print(…)` do?

What does `print(…)` *print*?



40

Self-check: What are the *values* in this code?

script.py

```
1  print("Congratulations!")
2  print("Your percentage was:")
3  print(100/100)
```

41

Values have different *types*

Example	Type	Description
"Hello, World" "Go Bengals" "My score is 99.5%" "I am 35 years old"	**string**	_____ _____
3 10000000 -45	**integer**	_____
3.1 -0.5 10000000.3415	**float**	_____

WARNING:
"35" is a string!
35 is an integer!

WARNING:
1,000 — NO
$35.43 — NO!
75% — NO!

42

TAPPS—Identify the *type* for each value

2.2

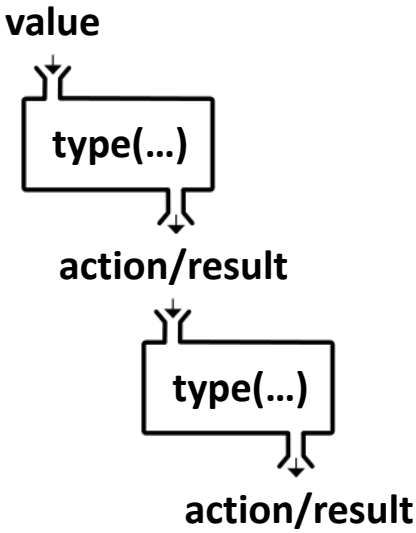
Value	Type
“You so totally rock.”	
0	
“45”	
59.65	
-1001	
0.75	
“Pizza is so delicious”	
10,004	
“$10,004.45”	

43

Use the `type(…)` function to find a value’s type

2.2

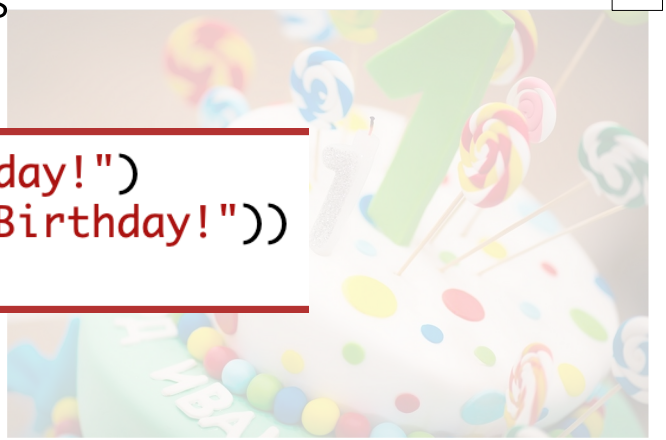
```
print(type("Hello, World!"))
print(type(17))
print("Hello, World")
```



Try it!
Run the code
in E-Book 2.2

44

TAPPS—**Read** this Python program and **describe** what it does



```
print("Happy Birthday!")
print(type("Happy Birthday!"))
type(17)
```

45

There are ***multiple*** ways to create strings


```
print('This is the Pythonic way.')

print("This is also common.")

print"""This way allows
line breaks."""

print'''Want "quotation" marks?'''

```

46

You can give ***multiple*** values to the `print(…)` function


```
print("I am",35,"years old")
```

Separate values with a _____

47

ice_cream.py: Write a program that prints out your favorite ice cream flavor, the number of scoops, and the price.


```
print("Mint chocolate chip")
print(3, "scoops")
print("$",5.50)
```

Code a little, debug as you go, keep a working program

48

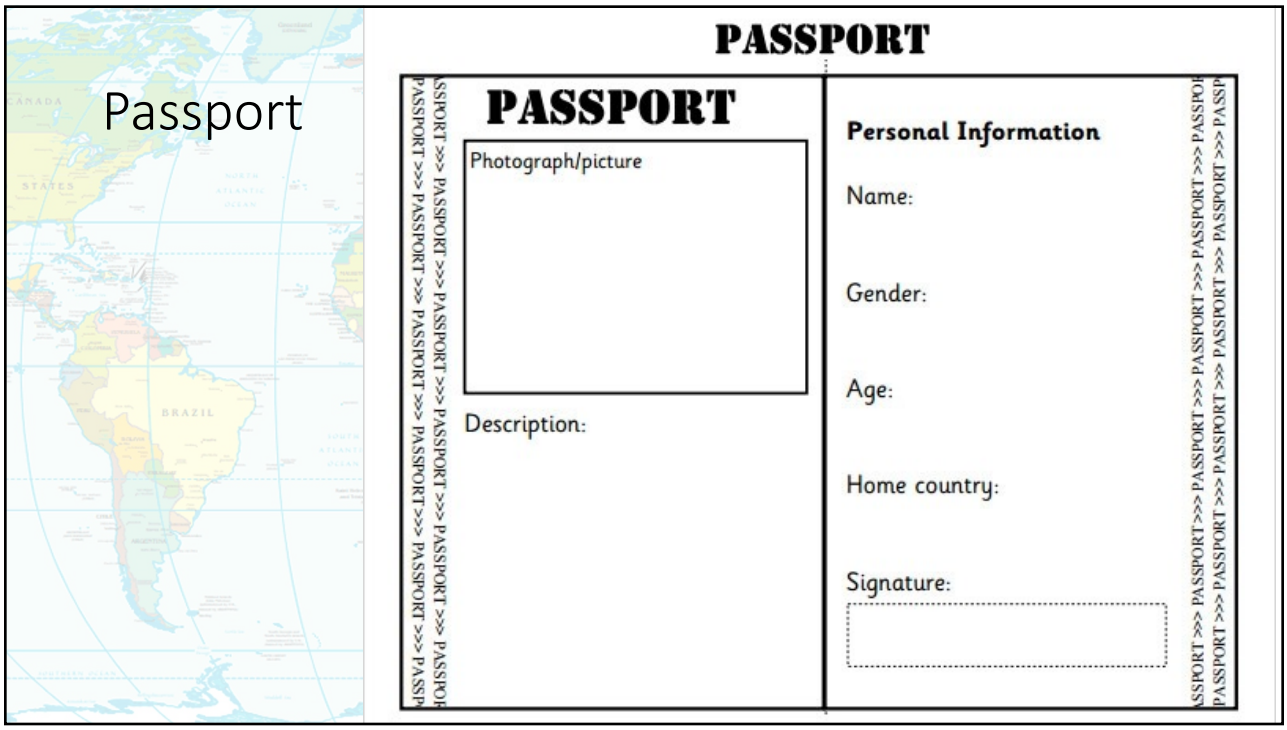
You will need to know how to convert types

If you have…	And you want…	Use…	Examples
float or **int**	**string**	str(…)	str(5) ->___ str(35.41) ->_____
float or **string**	**integer**	int(…)	int("100") -> ____ int(5.999) ->____
int or **string**	**float**	float(…)	float(20) ->____ float("0.123") ->____

`int(…)` does not _____, it _____

49

Passport



50

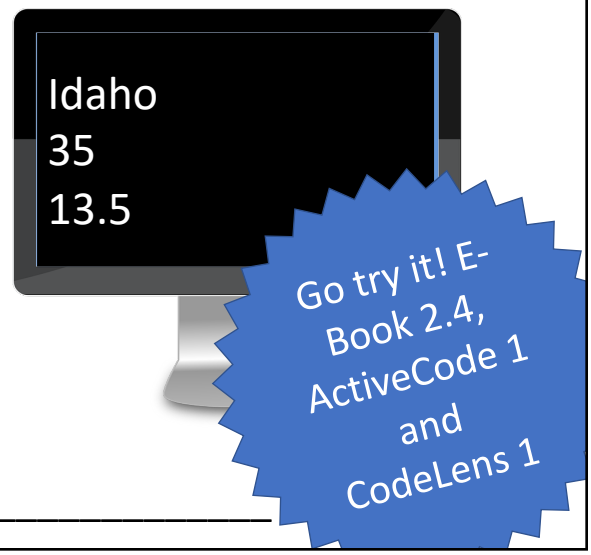
Values can be assigned to names

2.4

name

value

```
1 state = "Idaho"
2 age = 35
3 shoe_size = 13.5
4
5 print(state)
6 print(age)
7 print(shoe_size)
```

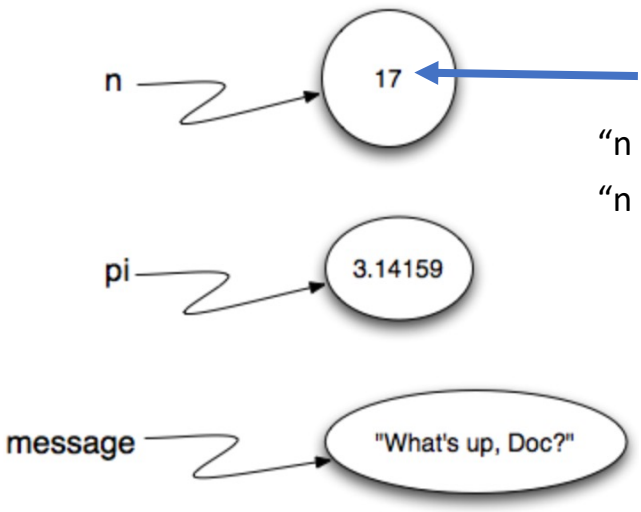


A ***name*** that holds a ***value*** is called a _____

51

How to think about **variables**

2.4



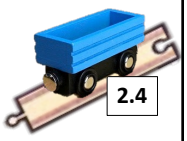
This value can change!

"n equals 17" — NO!

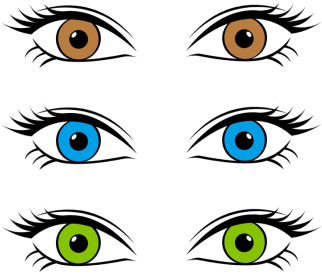
"n is _____ the value 17" —YES!

52

variables.py: Write a program that creates 3 variables for: your eye color, shoe size, and grade. Then print the variables in a nice way.

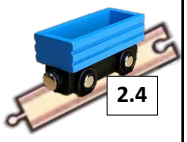



```
print("I have ", eye_color, "eyes")
```



53

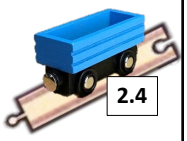
3 common mistakes with variables




```
first name = "Paul"
last_name == "Bodily"

print(last_Name)
```

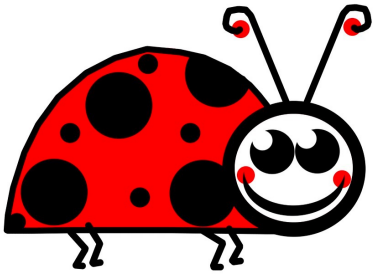
54

Identify the bugs in this code




```
favorite_movie = “The Majestic”
favorite color == “Green”

print(Favorite_movie)
print(favorite color)
```

- What kinds of errors are these?



55

RACE —

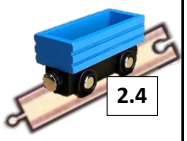
- Split in two groups. As a group you must code up a program that prints the following:

```
2 Sausage Burrito 3.58
1 Mild Picante 0.39
1 Small Water 0.05
```

- Each person can only add/edit one line and run once.
- Your program should use:
  - 9 variables (3 integers, 3 strings, and 3 floats)
  - 3 print statements that only print variables
  - No brown squiggly lines

56

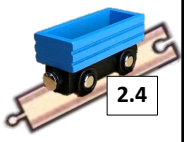
A variable has the same type as its value




```
state = "Idaho"
age = 35
shoe_size = 13.5

print(type(state))
print(type(age))
print(type(shoe_size))
```

57

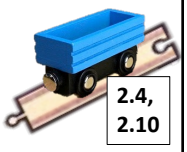
A variable is used to **remember** things




```
line1 = "Twinkle, twinkle, little star, how I wonder what you are"
line2 = "Up above the world so high, like a diamond in the sky"

print(line1)
print(line2)
print(line1)
```

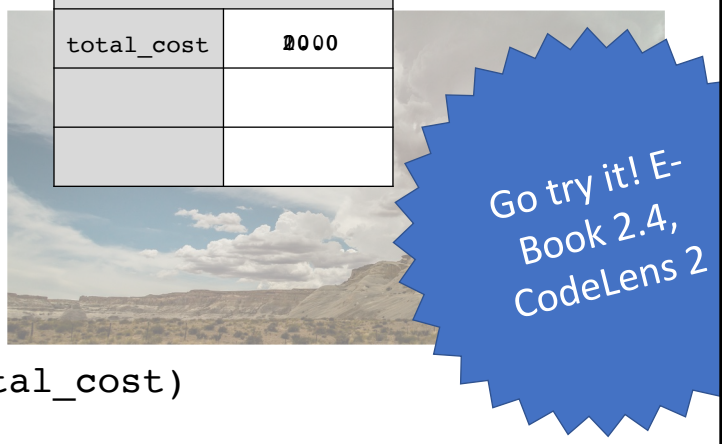
58

Variables can *change* their values




```
1 total_cost = 0.0
2 print(total_cost)
3
4 total_cost = 10.0
5 print(total_cost)
6
7 total_cost = 20.0
8 print("Total =",total_cost)
```

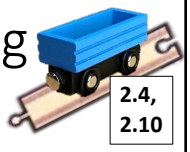
Variable Table	
total_cost	0.0 10.0 20.0



WARNING: Variables in _____ do not change their values

59

TAPPS—What is printed when the following statements execute?



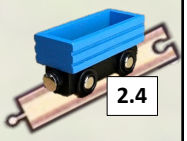

```
day = "Thursday"
day = 32.5
day = 19
print(day)
```

A. Nothing is printed. A runtime error occurs.
B. Thursday
C. 32.5
D. 19

Variable Table	
day	"Thursday" 32.5 19

60

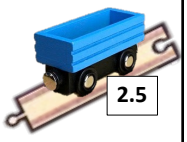
game_score.py—Write a program that has a variable called **`score`** whose starting value is 0. Print the variable **`score`**. Change the value of **`score`** and print it. Change and print **`score`** again.



Code a little, debug as you go, keep a working program

61

Some variable names are illegal



Variable names…

```
76trombones = "big parade"
more$ = 1000000
class = "Computer Science 101"
```

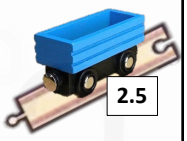
must start with _____

can only contain _____

cannot be _____

62

TAPPS—Identify which variable names are illegal and why




```
city state
winner!
class
_cost
grade%
1st_name
6th_grade_teacher
answer_1
count
print_size
```

63

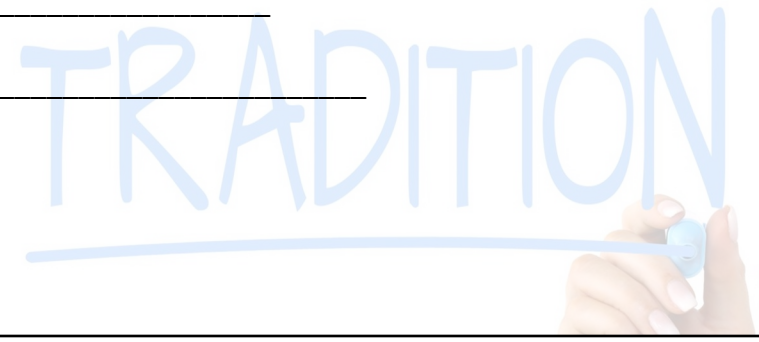
Variable names should follow *convention*

Variable names ***should***

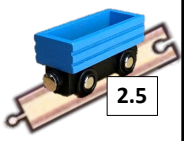
_____

_____

_____



64

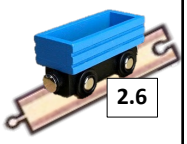


birthday.py—Using valid variable names write a program that creates variables for a 1) birth month, 2) birth date, and 3) birth year. Then print it out using the format "January 1, 2000"

Code a little, debug as you go, keep a working program

65

An **expression** is a combination of values, variables, operators, and calls to functions



A _____ is an instruction that Python can execute

```
1 print(1 + 1)
2 print(len("hello"))
3
```

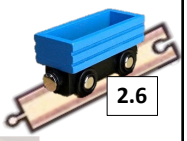
These are 2 ***statements***

These are 2 ***expressions***

An expression produces a _____

66

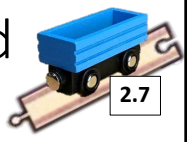
TAPPS—Identify the ***expressions*** and the ***values*** they produce




```
y = 3.14
x = len("hello")
print(x)
print(y)
print(x+y)
```


67

Operators (+,-,*,/) operate on values called ***operands***




```
20 + 32
hour - 1
hour * 60 + minute
minute / 60
5 ** 2
(5 + 9) * (15 - 7)
```

Go try it! E-Book 2.7, ActiveCode 1

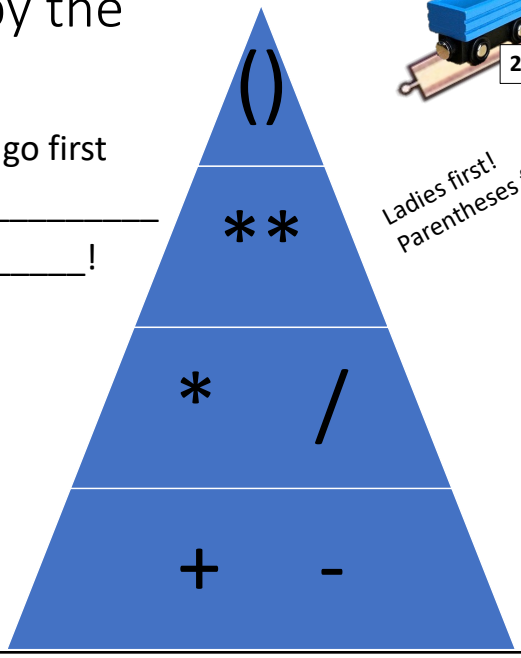
68

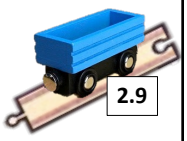
Operators are evaluated by the *Order of Operations*

Higher _____ go first

Equal precedence, go _____

To be clear and simple, _____!



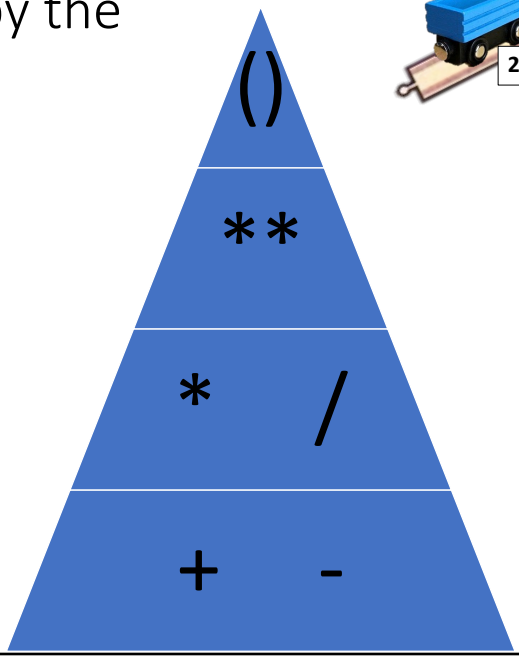


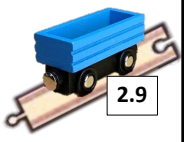
Ladies first!
Parentheses first!

70

Operators are evaluated by the *Order of Operations*

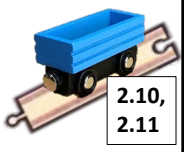
a) 2 * (3-1) -> __
b) (1+1)**(4-2) -> __
c) 2**1+1 -> __
d) 3*1**3 -> __
e) 2*3-1 -> __
f) 5-2*2 -> __
g) 6-3+2 -> __
h) 16 - 3 * 4 / 2 + 1 -> __





71

The ***result*** of the expression on the right of "=" becomes the ***value*** for the variable on the left




```
x =  15
y =  x + 5
x =  22
```

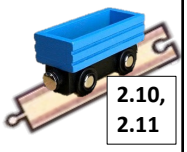
Only the variable on the _____ side changes value

```
z = 15
z = z + 5
```

To _____ a variable, put it on **both** sides

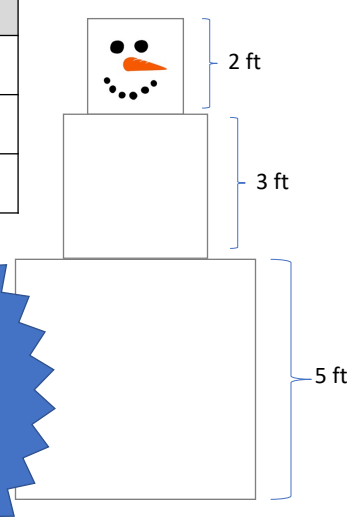
72

Computing the **surface area** of Sam, the square snowman




```
total_area = 0.0

# compute/add top square
width = 2.0
area = width * width
total_area = total_area + area

# compute/add middle square
width = 3.0
area = width * width
total_area = total_area + area

# compute/add bottom square
width = 5.0
area = width * width
total_area = total_area + area

print("Total area =",total_area)
```

Variable Table	
total_area	0.0 4.0 13.0 38.0
width	2.0 3.0 5.0
area	4.0 9.0 25.0



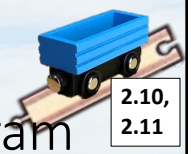
Find the variable table for your programs

73

BTW—you can "add" strings together to make new strings

```
month = "April"
day = 9
year = 1986

print(month, day, ",", year)

date = month + " " + str(day) + ", " + str(year)
print(date)
```

74



Live Coding demonstration—Write a program with variables representing the miles driven on 3 different days (you choose the miles). Create a 4th variable that is ***updated*** to reflect the total count of all 3 variables. Print out all four variables.

Code a little, debug as you go, keep a working program

75

store.py —Congratulations! You own a store selling your favorite things! Write a program with variables representing the cost of 3 specific items (you choose the cost). Create a 4th variable that is ***updated*** to reflect the total cost of all 3 items. Print out all four variables.

2.10, 2.11

Code a little, debug as you go, keep a working program

76

Live Coding demonstration—Problem-Solving:

2.10, 2.11

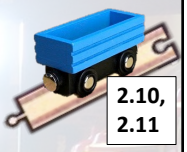
"Mom gives you $20. First, you buy a bus ticket for $1.50. Next, you buy pop corn for $6.25. Finally, you buy a movie ticket for $8.50."

Write a program that prints out each sentence in the story. After each sentence print out how much $ you have left. Use variables to represent the 4 values in the story.

Code a little, debug as you go, keep a working program

77

restaurant.py—Problem-Solving:



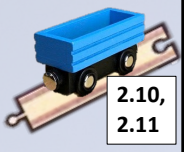
"You've just finished eating at Pizza Hut with 2 friends. The waiter brings you the bill for $21.50. You decide to split the bill."

Write a program that creates 3 variables for: the name of the restaurant, the total bill, and the number of people sharing the bill. Print out these three values and also how much each person will pay.

Code a little, debug as you go, keep a working program

78

gas_station.py — Problem-Solving:



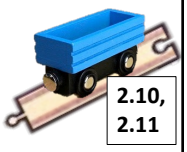
"You are at the gas station filling up. You pump 9.3 gallons of gas. The price per gallon is $2.89. You also buy a slurpee for $1.39."

Write a program to calculate and print A) the cost of just the gas and B) the total cost. Use a variable `running_total` for both A and B.

<u>Bonus</u>: Compute and print the total cost after 6% sales tax.

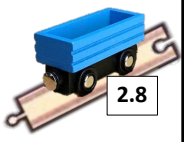
79

2-minute write: What have you learned so far about programming? Or Write 3 questions you have
Questions?



80

TAPPS—***Read*** the program below (***describe*** each line in detail). ***Summarize*** what the program is doing.




```
pounds_per_dollar = 0.79   # exchange rate
dollar_amount = 135.00
print("Exchange Rate: $1 = £",pounds_per_d
print()

print("You have $",dollar_amount)
print()
# convert dollars to pounds
pounds_amount = dollar_amount * pou
print("That equals £", pounds_amoun
```

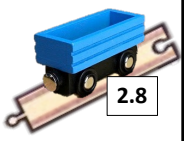
Exchange Rate:
$1 = £ 0.79

You have $ 135

That equals £ 106.65

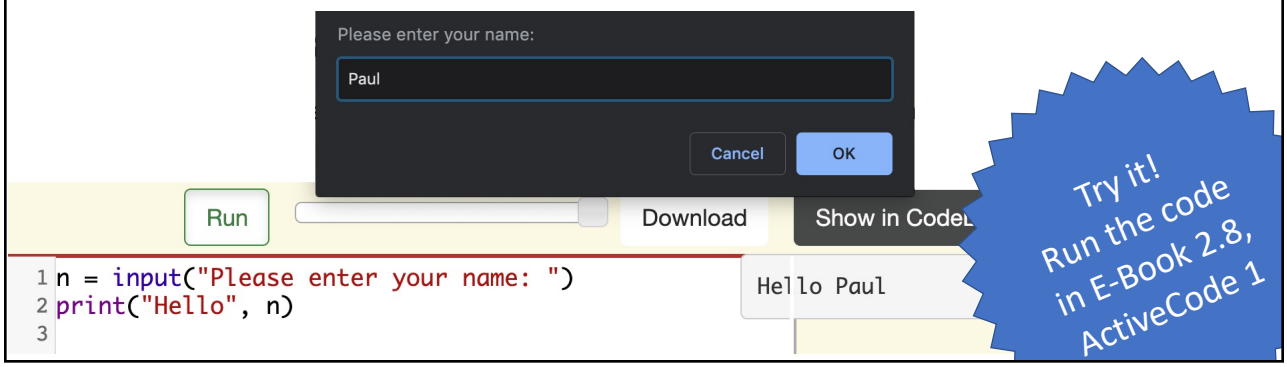
81

Use `input(…)` to get a value from the user



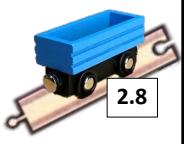

```
result = input("Put a prompt here")
```

`input(…)` always returns a _____!

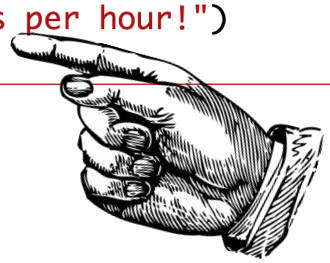


82

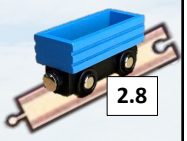
TAPPS—Identify all the variables in the following program. For each variable identify its data type.




```
str_wpm = input("How many words per minute can you type?")
int_wpm = int(str_wpm)

# calculate words per hour from words per minute
wph = int_wpm * 60

print("Then you can type", wph, "words per hour!")

```



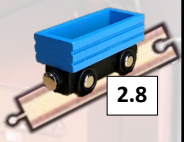
83



Live Coding demonstration—Copy and adapt your program summing the mileage so that the 3 daily mileage variables are set by user input.

Code a little, debug as you go, keep a working program

84



store_input.py — Your store (selling your favorite things) is thriving! Copy and adapt your program to let the user input prices for the 3 items.

Code a little, debug as you go, keep a working program

85

Live Coding demonstration—Problem-Solving:

2.8

"Mom gives you money. First, you buy a bus ticket. Next, you buy popcorn. Finally, you buy a movie ticket."

Copy and adapt your previous program to allow the user to input the different values. After each sentence print out how much $ you have left.

Code a little, debug as you go, keep a working program

86

restaurant_input.py — Problem-Solving:

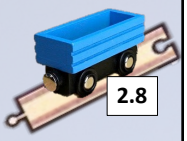
2.8

"You've just finished eating at Pizza Hut with friends. The waiter brings you the bill. You decide to split the bill."

Copy and adapt your previous program to let the user input the name of the restaurant, the total bill, and the number of people sharing the bill.

Code a little, debug as you go, keep a working program

87



gas_station_input.py—Problem-Solving:

"You are at the gas station filling up. You pump the gas. You observe the price per gallon. You also buy a slurpee."

Copy and adapt your previous program to let the user input all of the values.

Code a little, debug as you go, keep a working program

88

cave_story.py: Write a Python program that 1) prints the text below, 2) prompts the user, 3) prints the users choice

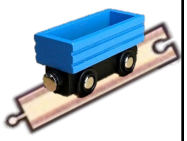
```
You reach a cave. You can either:
    A. Go in the cave
    B. Shout into the cave
    C. Wait outside
What do you choose?                <-prompt
You chose _____.                   <-print the result
```

"This character \n puts in a _____"

"This character \t puts in a _____"

90

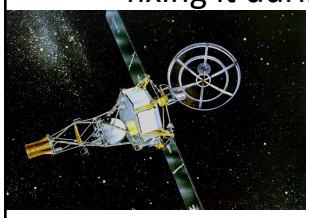
2-minute pause: What has been A) the coolest thing you've learned and B) the most difficult part of the course so far?

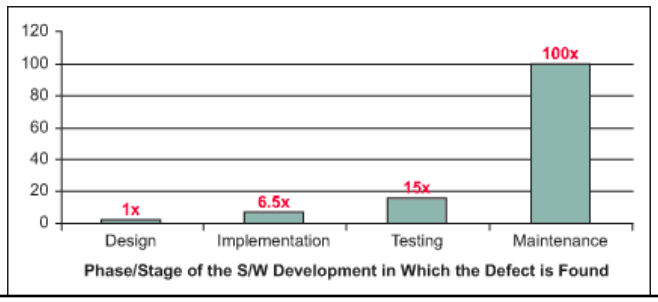


91

TAPPS—Come up with an answer and some explanation for why you chose it:

- The average professional programmer makes _____ errors per 1000 lines of delivered code. (Code Complete)
- In 2016, software bugs cost $_____ worldwide and affected _____ customers. (crossbrowsertesting.com)
- Fixing a bug after the program is released costs _____ times more than fixing it during design. (IBM)





92

Tricks to avoid lots of debugging

- **Identify a _____ solution**
  - **Start _____**
    - **Keep _____ it**

Let's look at Section 3.2 together

93

More tricks to avoid frustration

Rule #1: _____ is not the problem

Rule #2: _____
and _____ are clues

94

Message	Number	Percent
ParseError:	4999	54.74%
TypeError:	1305	14.29%
NameError:	1009	11.05%
ValueError:	893	9.78%
URIError:	334	3.66%
TokenError:	244	2.67%
SyntaxError:	227	2.49%
TimeLimitError:	44	0.48%
IndentationError:	28	0.31%
AttributeError:	27	0.30%
ImportError:	16	0.18%
IndexError:	6	0.07%

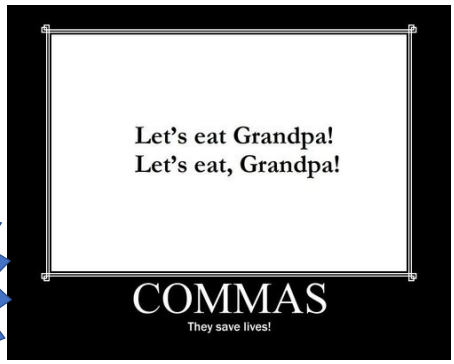
95

ParseError means you _____ a comma, a parenthesis, or something else.

- how do you know where this sentence starts is a mystery without punctuation you don't know how should i interpret what you mean

```
print "hello world")
x = "apple"3
```

ActiveCodes 2 & 3 together in Section 3.4.1



96

TypeError means you have _____ *data types*

- What do you get when you add an integer and a string?

```
x = "banana" + 3
```

ActiveCode 4 together in Section 3.4.2

97

NameError means you used a variable that _____ got a value

- "I'm having the repairman come fix my clinkerbot."

```
first_name = "Paul"
print(firstname)
```

ActiveCodes 5 & 6 together in Section 3.4.3

98

ValueError means you put a value in a function that _____ the function

3.4.4

Turn the word “melon” into a number.

```
count = int(“melon”)
```

ActiveCode 8 together in Section 3.4.4

99

mad_lib.py: Create a MadLib

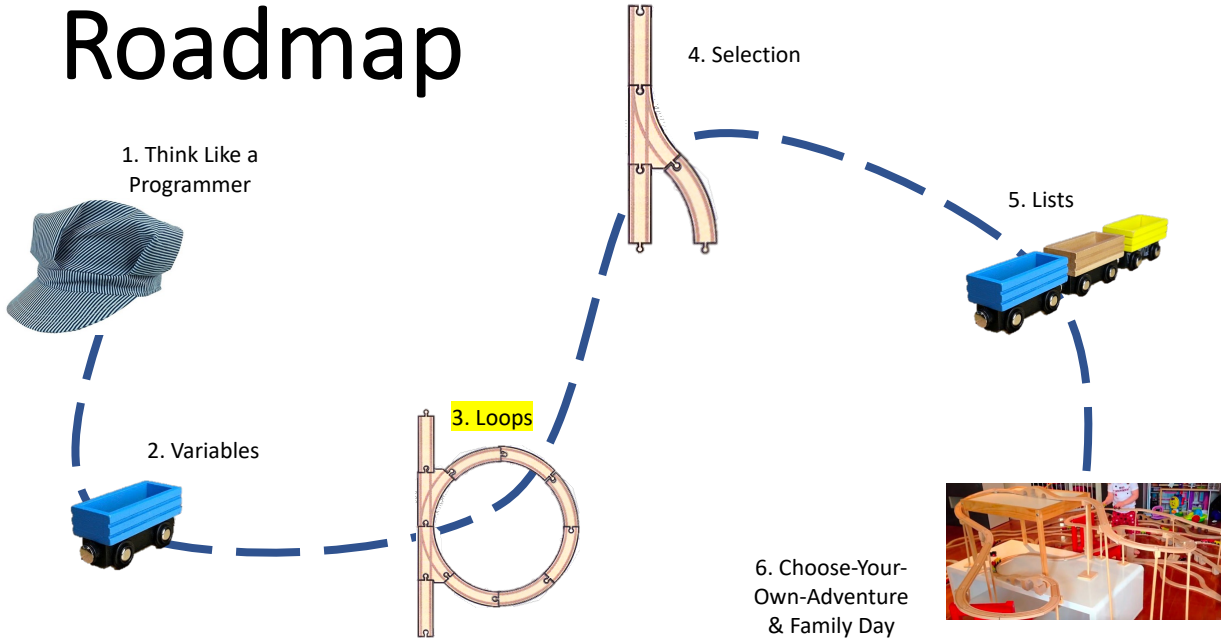
1. Rewrite a 3-4 sentence version of a classic fairy tale.
2. Replace 3-4
   - Nouns – person/place/thing
   - Adjectives – words that describe
   - Verbs – action words
   - Numbers (at least 1 int and 1 float)
3. Prompt the user to input these 3-4 words, storing each in a separate variable.
4. Then print out the story with the spaces filled in.

Start small, code a little, debug as you go

It was a _____ , cold November day. I
adjective
woke up to the _____ smell of _____
adjective type of bird
roasting in the _____ downstairs. I
room in a house
_____ down the stairs to see if I could
verb (past tense)
help _____ the dinner. My mom said,
verb
"See if _____ needs a fresh _____." So I
relative's name noun
carried a tray of glasses full of _____ into
a liquid
the _____ room. When I got there, I
verb ending in -ing
couldn't believe my _____! There were
part of the body (plural)
_____ _____ on the _____!
plural noun verb ending in -ing noun

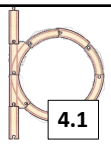
100

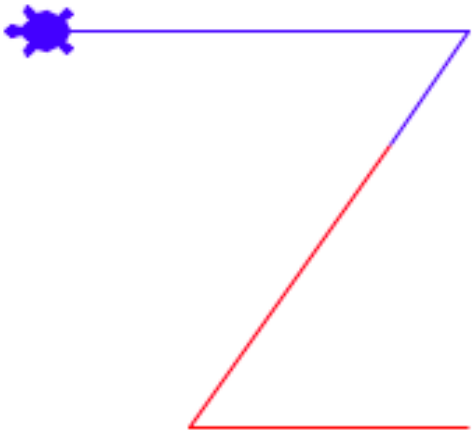
Roadmap

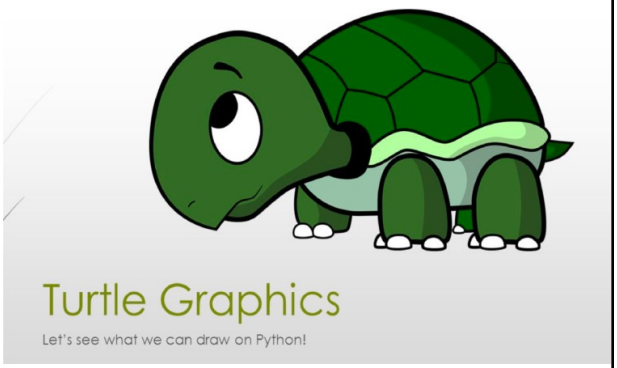


101

Turtle graphics lets us output drawings instead of text







102

Python can **import** *modules* that do cool stuff

4.2

```
import turtle                # allows us to use the turtles library
wn = turtle.Screen()         # creates a graphics window
alex = turtle.Turtle()       # create a turtle named alex
alex.forward(150)            # tell alex to move forward by 150 units
alex.left(90)                # turn by 90 degrees
alex.forward(75)             # complete the second side of a rectangle
```

Complete yellow-box activities to AC2 for section 4.2

...tle starts facing _____

103

Use a **for** *loop* to repeat code over and over again

4.4

I am a _____

```
for var_name in [val1, …, valN]:
    statement1
    statement2
```


```
print("Hello, world!")

for name in ["Joe", "Amy", "Brad", "Angelina", "Zuki", "Thandi", "Paris"]:
    print("Hi", name, "Please come to my party on Saturday!")

print("Done!")
```

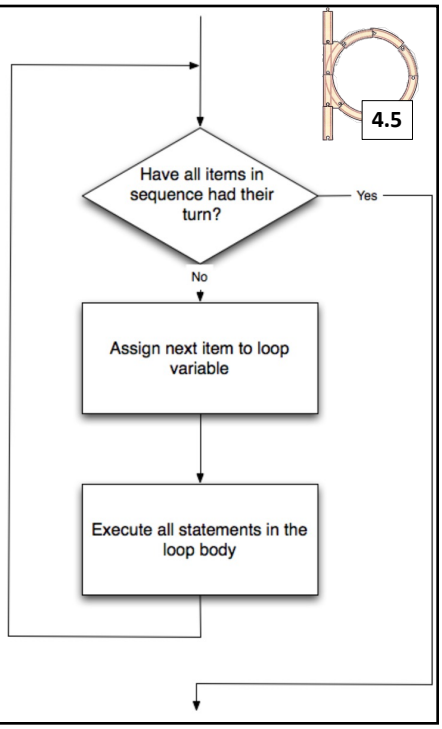
Statements in a `for` loop have the same _____

104

A `for` loop **controls** the *flow* of the program

4.5

```
print("Hello, world!")

for name in ["Joe", "Amy", "Brad", "Angelina", "Zuki", "Thandi", "Paris"]:
    print("Hi", name, "Please come to my party on Saturday!")

print("Done!")
```



CodeLens for Section 4.5 Phanon

105

Describe the flow of execution in this program

4.5

```
print("Hey, friend!")

for day in ["Monday", "Wednesday", "Friday"]:
    place = input("Where should we meet on " + day + "? ")
    print("I will meet you at", place, "on", day)

print("It's a plan, Stan!")
```

106

Describe the flow of execution in this program

4.5

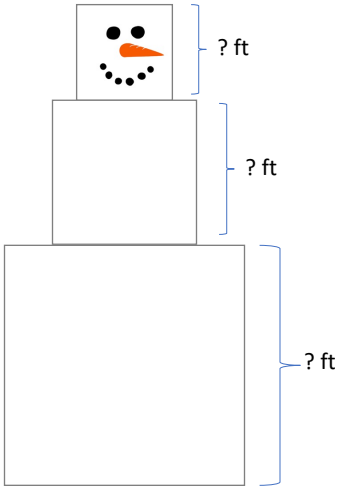
```
friend_count = 0

for friend in []:
    friend_count = friend_count + 1

print("Paul has", friend_count, "friends!")
```

107

```
total_area = 0.0

# compute/add top square
width = 2.0
area = width * width
total_area = total_area + area

# compute/add middle square
width = 3.0
area = width * width
total_area = total_area + area

# compute/add bottom square
width = 5.0
area = width * width
total_area = total_area + area

print("Total area =",total_area)
```

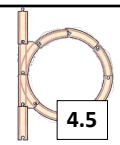
TAPPS! How could you use a loop to simplify this code?

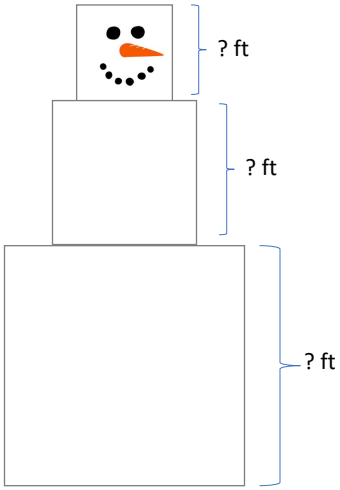
4.5



108

```
total_area = 0.0

# Repeat these instructions 3 times for
# 3 different widths:
for width in [2.0, 3.0, 5.0]:
    # compute/add top square
    area = width * width
    total_area = total_area + area

print("Total area =",total_area)
```





109

Live Coding Demo—Write a program that uses a for loop to print

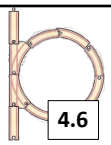
```
One of the months of the year is January
One of the months of the year is February
One of the months of the year is March
```

etc …

Comment often and clearly

110

turtle_square.py—How would you **modify** this code using a for loop to draw a square?




```
import turtle                # set up alex
wn = turtle.Screen()
alex = turtle.Turtle()

alex.forward(50)
alex.left(90)

wn.exitonclick()

```

111

Live Coding Demo—

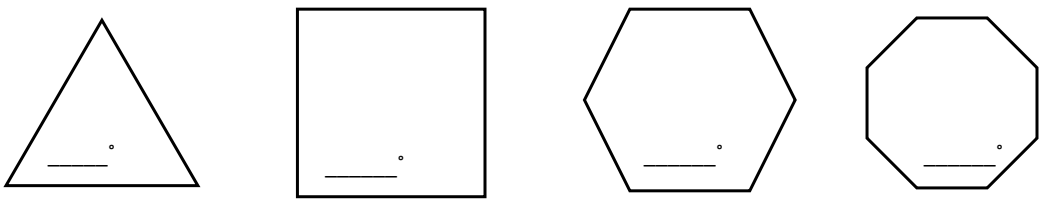
A drunk pirate makes a random turn and then takes **100 steps forward**, makes another random turn, takes another 100 steps, turns another random amount, etc. A social science student records the angle of each turn before the next 100 steps are taken. Her experimental data is **160, -43, 270, -97, -43, 200, -940, 17, -86**. (Positive angles are counter-clockwise.) Use a turtle to **draw the path** taken by our drunk friend. After the pirate is done walking, **print the current heading**.

Comment often and clearly

112

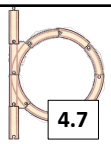
turtle_shapes.py—

Use `for` loops to draw these polygons (all sides the same lengths, all angles the same):



113

Use the `range(…)` function for *common* `for` loops



- range(…) generates _____

```
import turtle
wn = turtle.Screen()
alex = turtle.Turtle()

for i in [0, 1, 2, 3]:
    alex.forward(50)
    alex.left(90)

wn.exitonclick()
```

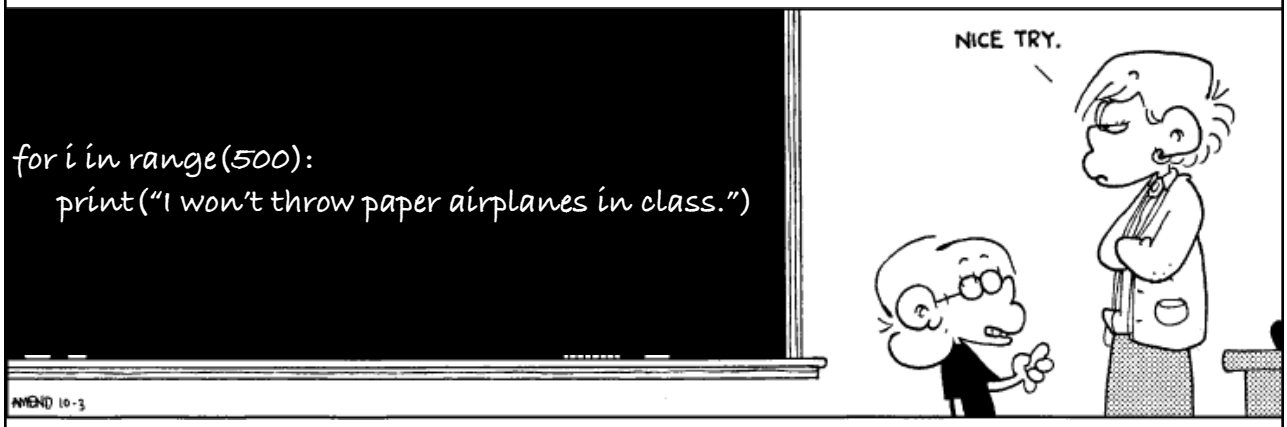
is the same as

```
import turtle
wn = turtle.Screen()
alex = turtle.Turtle()

for i in range(4):
    alex.forward(50)
    alex.left(90)

wn.exitonclick()
```

114

Describe the flow of execution in this program

4.7



115

square_numbers.py—Write a program that prints A) the numbers 0 to 100 and B) their squares. Use a `for` loop with the `range`(…) function.

4.7

For example:

0	0
1	1
2	4
3	9
4	16
5	25
…	

Comment often and clearly

Code a little, debug as you go, keep a working program

116

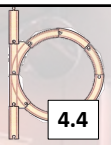
story_loop.py: Write a program that prints out the following (or something similar)

```
Welcome to Treasure Island! You have scurvy! You have
10 days (moves) to find fruit before you die.

Day 0 (you have 10 days left)
Day 1 (you have 9 days left)
…
Day 9 (you have 1 days left)

Day 10. You died of scurvy!
```

***Start simple**, code a little, debug*

117

store_loop.py—Your store (selling your favorite things) is *thriving*! Look back at your store.py program: how could you use a for loop to add up the cost of multiple items?



Write a new program that uses a **`for`** loop (with **`range(…)`** function) to sum the cost of the 3 items.

Then enhance the program to let the *user* input how many items they are buying.

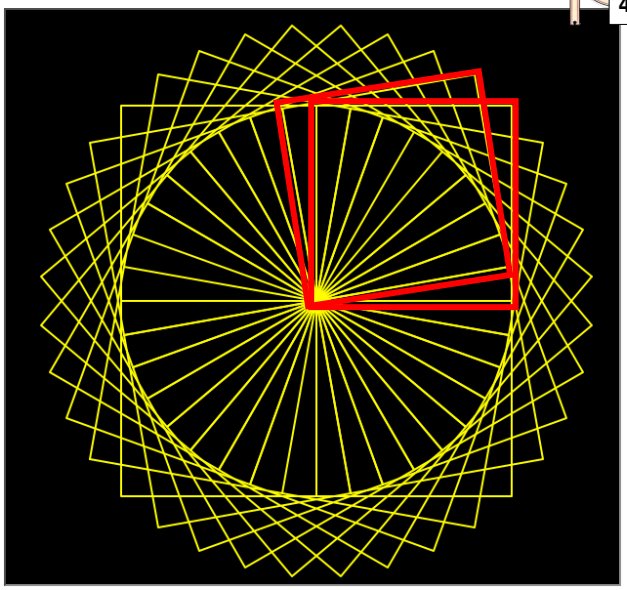
Comment often and clearly

Code a little, debug as you go, keep a working program

118

Live Coding Demo—

- How would you draw this picture with turtle graphics?

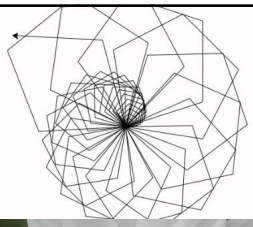


4.7

Comment often and clearly

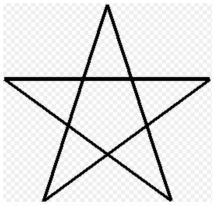
119

Project:Using a for loop and the range(…) function write a program to draw some kind of **awesome picture**. Be creative and experiment with the turtle methods provided in sections **4.8 and 4.9**.



4.8

Comment often and clearly



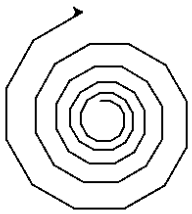
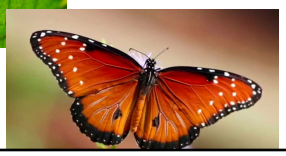


Figure 1c



120

Using and reassigning variables in a `for` loop

```
# starting flavor
flavor = "vanilla"

for i in range(3):
      print("You tried", flavor)
      flavor = input("What would you like next?")
```

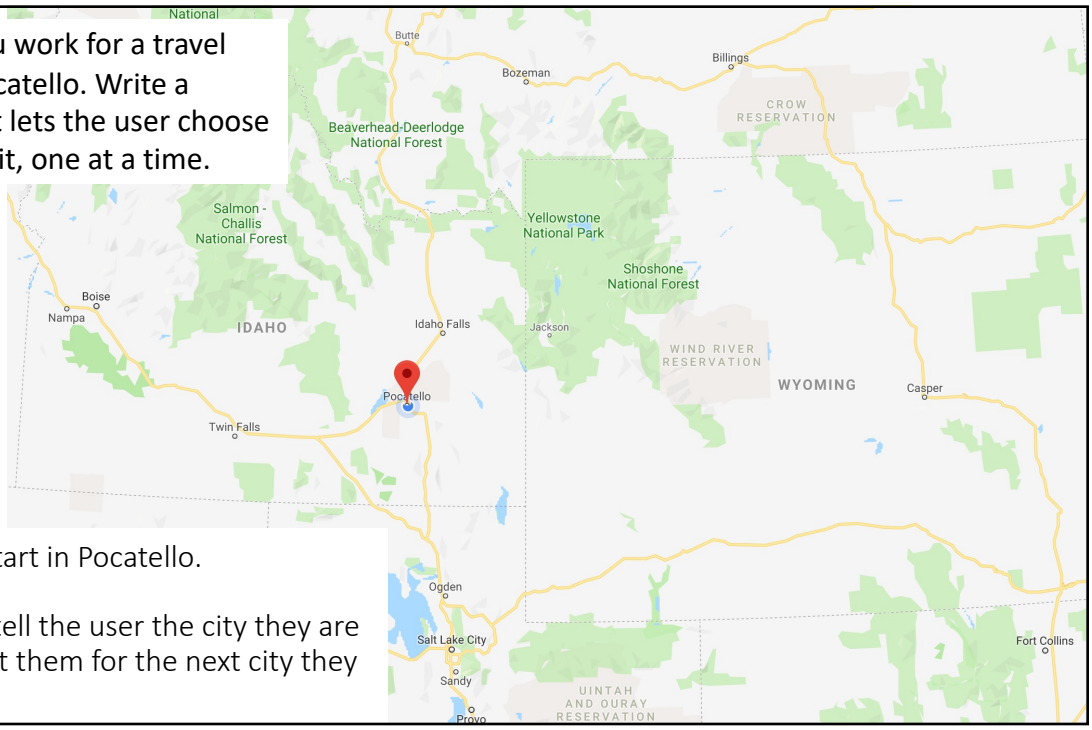
The variable value at the _____ of one loop will be its value at the _____ of the next

```
You tried vanilla
What would you like next? chocolate
You tried chocolate
What would you like next? mint
You tried mint
What would you like next? strawberry
```

Variable Table	
flavor	~~chocolate~~ ~~mint~~ ~~strawberry~~
i	~~0~~ ~~1~~ ~~2~~

121

travel.py: You work for a travel agency in Pocatello. Write a program that lets the user choose 5 cities to visit, one at a time.



They should start in Pocatello.

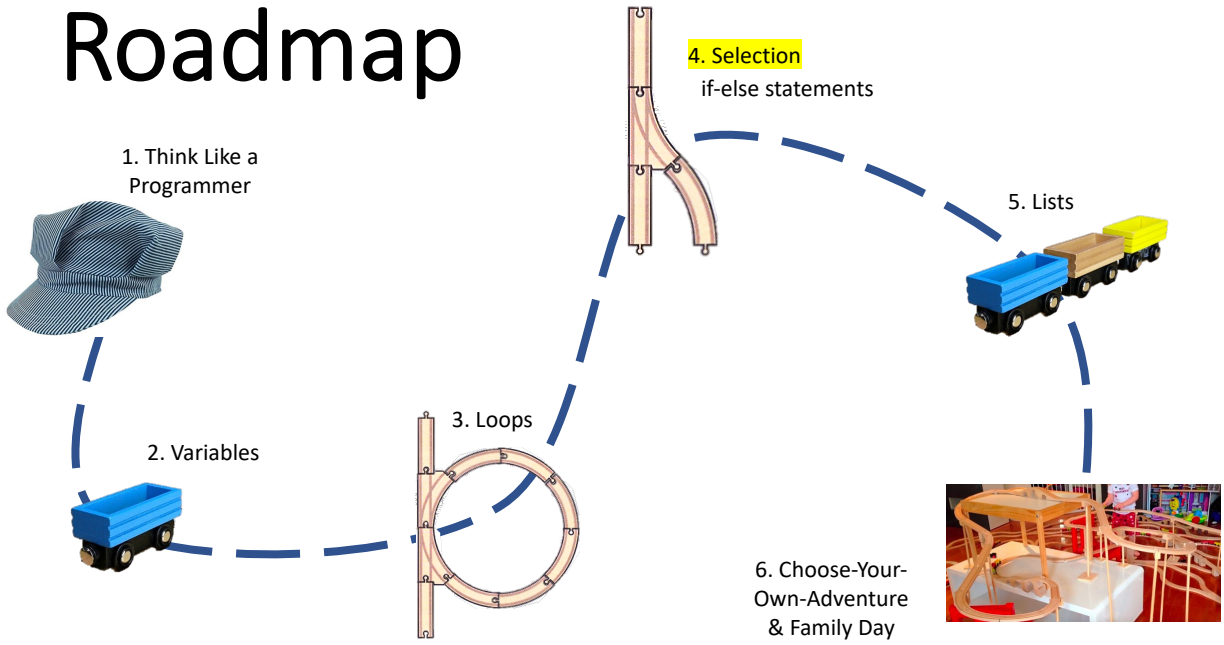
At each step, tell the user the city they are in, and prompt them for the next city they want to visit.

122

2-minute write: What do you understand about for loops?

123

Roadmap



124

TAPPS—Twitter Sports Winner Announcer

7.1

- Design (i.e., in comments) a program that prompts the user for two integers: Team A's score and Team B's score. Then print out a message that declares and congratulates the winning team.

```
team_A_score = input("Team A's score:")
team_B_score = input("Team B's score:")

# if team_A_score is bigger than team_B_score
print("Congratulations Team A!")
# otherwise
print("Congratulations Team B!")
```

125

When you should use if

7.1

If the solution uses _____,

use an _____ statement in the code

You must identify the _____

126

boolean *is a type with only two possible values:* ***True, False***

7.1

Example	Type	Description
"Hello, World" "My score is 99.5%"	**string**	**words**, a string of characters
3 -45	**integer**	**whole numbers**
-0.5 10000000.3415	**float**	**decimal** numbers
True False	**boolean**	_____ _____

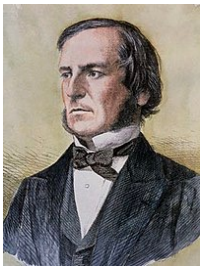
127

boolean is a data type like string, integer, and float

7.1

```
print(True)
print(type(True))
print(type(False))
```

```
True
<class 'bool'>
<class 'bool'>
```



George Boole

_____ is ***important***

Booleans are ***not*** _____

128

A ***boolean expression*** is an ***expression*** that evaluates to a boolean

7.1

An expression looks like this:

```
print((4*5) +1)
```

```
21
```

A boolean expression looks like this

```
print(5 == 5)
print(5 == 6)
```

```
True
False
```

I am a **comparison operator**

Comparison Operator	Meaning	Example
x == y	*True if x and y are* _____	5 == 5 (True) 5 == 6 (False)
x != y	*True if x and y are* _____	5 != 6 (True) 5 != 5 (False)
x > y	*True if x is* _____ *than y*	5 > 4 (True) 5 > 6 (False)
x < y	*True if x is* _____ *than y*	5 < 6 (True) 5 < 4 (False)
x >= y	*True if x is* _____*than* *or* _____ *to y*	5 >= 4 (True) 5 >= 5 (True) 5 >= 6 (False)
x <= y	*True if x is* _____*than* *or* _____ *to y*	5 <= 6 (True) 5 <= 5 (True) 5 <= 4 (False)

129

Live Coding—Simple Number Guessing Game

7.1

- Design and write a program that assigns a number (chosen by you) between 1 and 10 to a variable and then prompts the user to guess the number.
- Print out "True" or "False" based on whether the guessed number equaled the number you assigned.
- Adapt the program to give the user 5 guesses.

1 2 3 4 5 6 7 8 9 10

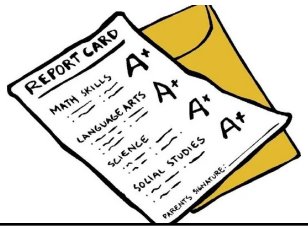
130

pass_fail.py—Pass/Fail Program

7.1

You are teaching a pass/fail class. Passing means a student got a score of **70 or higher**. Design and write a program that prompts the user for a score between 0 and 100.

Print out "True" or "False" based on whether the student passed the class.

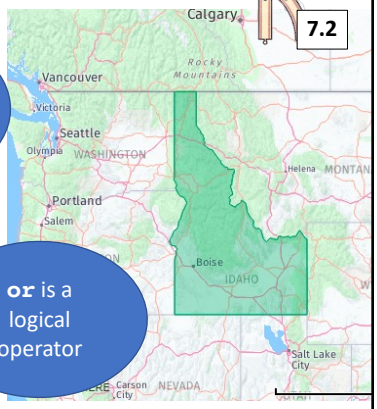


Comparison Operator	Example
x == y	5 == 5 (True) 5 == 6 (False)
x != y	5 != 6 (True) 5 != 5 (False)
x > y	5 > 4 (True) 5 > 6 (False)
x < y	5 < 6 (True) 5 < 4 (False)
x >= y	5 >= 4 (True) 5 >= 5 (True) 5 >= 6 (False)
x <= y	5 <= 6 (True) 5 <= 5 (True) 5 <= 4 (False)

131

booleans are combined using *logical operators*

7.2



What are valid ages for a *teenager*?

```
age = 12
print(age >= 13 and age <= 19)
```

and is a logical operator

What are valid area codes for *Idaho*?

```
area_code = 286
print(area_code == 208 or area_code == 986)
```

or is a logical operator

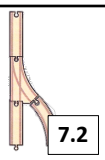
You can also flip a boolean value with `not`:

```
print(not (age >= 13 and age <= 19))
```

not is a logical operator

132

TAPPS—Complete Questions 1 and 2 in Section 7.10 of the e-Book



What do these expressions evaluate to?

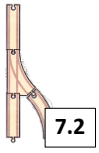
1. `3 == 3`
2. `3 != 3`
3. `3 >= 4`
4. `not (3 < 4)`

Give the **logical opposites** of these conditions. You are not allowed to use the `not` operator.

1. `a > b`
2. `a >= b`
3. `a >= 18 and day == 3`
4. `a >= 18 or day != 3`

133

TAPPS—Read and describe the program below




```
birth_year = int(input("Birth year:"))

print("To say you were born in the 80's is:")
print(birth_year >= 1980 and birth_year <= 1990)
```

Did you catch the error?

134

input_check.py—Checking User Input

7.2

- Design and write a program that prompts the user for a date in January.
- Print out “True” or “False” based on whether the input number is a valid date in January (hint: there is no January 32 and no January -1).

135

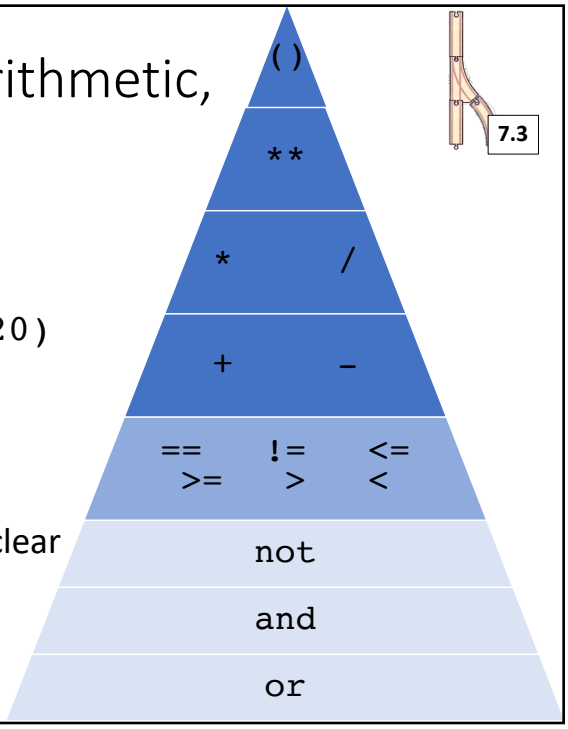
The order of operators is Arithmetic, Relational, Logical (ARL)

7.3

```
x = 2
y = 10
print(x*5 >= 10 and y-6 <= 20)
```

Use _____ and

_____ to be clear



136

TAPPS

7.3

Which of the following properly expresses the precedence of operators (using parentheses) in the following expression:

```
5*3 > 10 and 4+6==11
```

```
A. ((5*3) > 10) and ((4+6) == 11)
B. (5*(3 > 10)) and (4 + (6 == 11))
C. ((((5*3) > 10) and 4)+6) == 11
D. ((5*3) > (10 and (4+6))) == 11
```

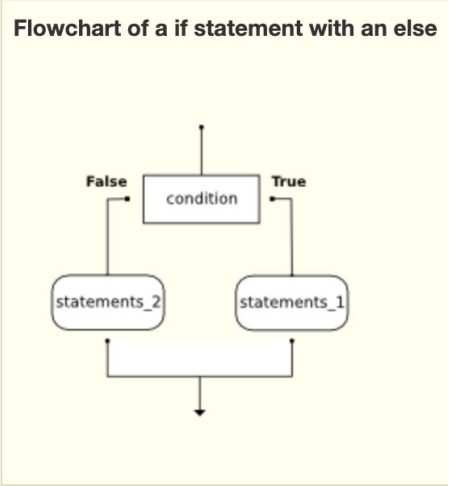
137

If-else statements use boolean expressions

7.4

```
x = 15

if x >= 0:
    print(x, "is positive")
else:
    print(x, "is negative")

```

```
if BOOLEAN EXPRESSION:
    STATEMENTS_1        # executed if condition evaluates to True
else:
    STATEMENTS_2        # executed if condition evaluates to False
```



138

Check Your Understanding

7.4

How many statements can appear in each block (the `if` and the `else`) in a conditional statement?

A. Just one.

B. Zero or more.

C. One or more.

D. One or more, and each must contain the same number.

139

Check Your Understanding

7.4

What does the following code print?

```
if 4 + 5 == 10:
    print("TRUE")
else:
    print("FALSE")
```

A. TRUE
B. FALSE
C. TRUE on one line and FALSE on the next
D. Nothing will be printed

140

Check Your Understanding

7.4

What does the following code print?

```
if 4 + 5 == 10:
    print("TRUE")
else:
    print("FALSE")
print("TRUE")
```

141

choose_winner.py—Twitter Sports Winner Announcer

7.4

- Design (i.e., in comments) a program that prompts the user for two integers: Team A's score and Team B's score. Then print out a message that declares and congratulates the winning team.

```
team_A_score = int(input("Team A's score:"))
team_B_score = int(input("Team B's score:"))

    print("Congratulations Team A!")

    print("Congratulations Team B!")
```

142

You can use `if` without `else`

7.5

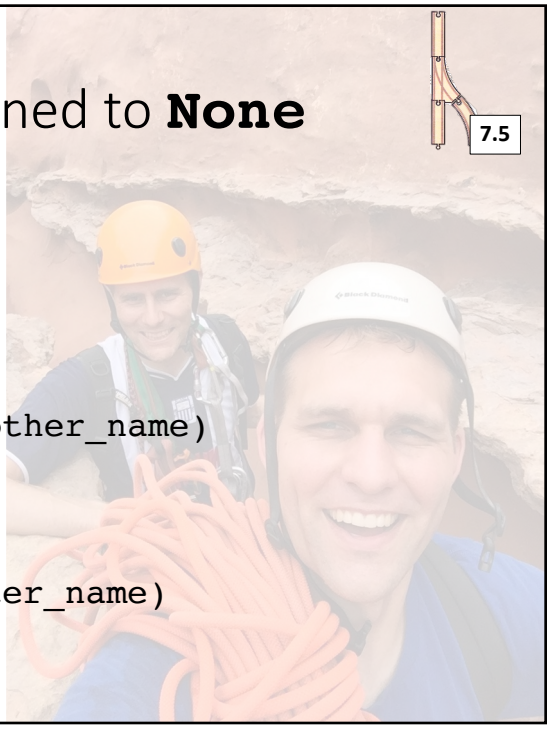
- This is common for _____

```
x = 10
if x < 0:
    print("The negative number ",  x, " is not valid here.")
    exit()
print("This is always printed")
```

143

A variable can also be assigned to **None**

7.5



```
brother_name = “Jason”
sister_name = None

# Check if brother has a name
if brother_name != None:
     print(“My brother is”, brother_name)

# Check if sister has a name
if sister_name != None:
     print(“My sister is”, sister_name)
```

144

drivers_license.py—Write a program that creates variables for 1) your name, 2) your state and 3) your driver's license number (if you don't have one, put **`None`**). Then print each variable, checking first to see if it is **`None`**. Assign these variables directly rather than using the **`input(…)`** function.

7.5

145

restaurant_zero_check.py—Problem-Solving:

7.5

"You've just finished eating at Pizza Hut with friends. The waiter brings you the bill. You decide to split the bill."

Copy and adapt your previous program to check if the number of friends is 0. If it is, print an error message and **`exit()`**.

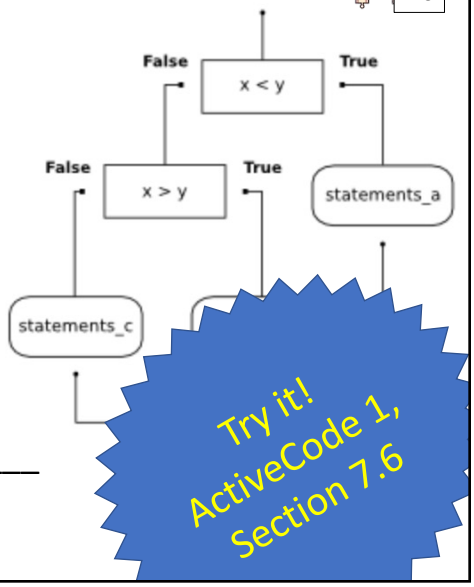
Do the same if the bill is less than 0.

146

If-else statements can be ***nested***

7.6

```
if x < y:
    print("x is less than y")
else:
    if x > y:
        print("x is greater than y")
    else:
        print("x and y must be equal")
```



A common mistake is

_____

Try it! ActiveCode 1, Section 7.6

147

Live Coding Demo—**Driver's license**! Write a program to prompt a user for their age. Then print out what kind of license they can apply for. These are the **minimum ages** for the following types of licenses:

7.6

0 — No license
14.5 — Supervised Instruction Permit
15 — Underage Driver's License
18 — Unrestricted Driver's License

148

movie_nested.py—Movie night! Prompt the user for their age. Then print the cost of a movie ticket based on their age.

7.6

- Ages 3 and under — FREE
- Ages 4 to 12 — $8.00
- Age 60 and over — $7.50
- General Admission — $10.00

149

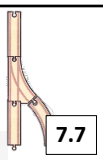
Live Coding—Adapt the program to account for College/Military IDs and Matinee showings

7.6

- Ages 3 and under — FREE
- Ages 3 to 12 — $8.00
- Age 60 and over — $7.50
- General Admission — $10.00

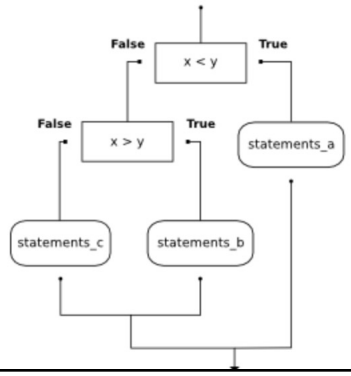
150

If-else statements can be *chained*



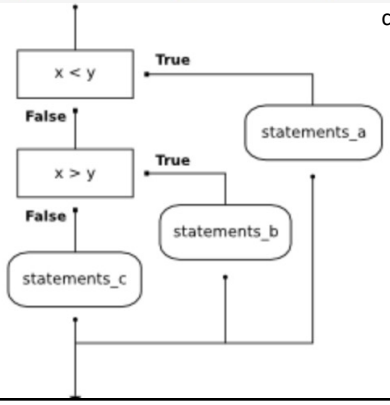

```
if x < y:
    print("x is less than y")
else:
    if x > y:
        print("x is greater than y")
    else:
        print("x and y must be equal")
```

nested




```
if x < y:
    print("x is less than y")
elif x > y:
    print("x is greater than y")
else:
    print("x and y must be equal")
```

chained



151

TAPPS—Answer the following question

select-7-2: What will the following code print if x = 3, y = 5, and z = 2?

```
if x < y and x < z:
    print("a")
elif y < x and y < z:
    print("b")
else:
    print("c")
```

- A. a
- B. b
- C. c

152

Live Coding Demo—**Driver's license**! Adapt your previous program to use ***chained*** if-else statements:

7.7

0 — No license
14.5 — Supervised Instruction Permit
15 — Underage Driver's License
18 — Unrestricted Driver's License

153

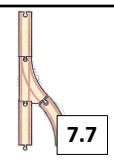
movie_chained.py—Movie night! Adapt your previous program to use ***chained*** if-else statements

7.7

- Ages 3 and under — FREE
- Ages 3 to 12 — $8.00
- Age 60 and over — $7.50
- General Admission — $10.00

154

TAPPS—Which of I, II, and III below gives the same result as the following nested if?




```
# nested if-else statement
x = -10
if x < 0:
    print("The negative number ",  x, " is not valid here.")
else:
    if x > 0:
        print(x, " is a positive number")
    else:
        print(x, " is 0")
```

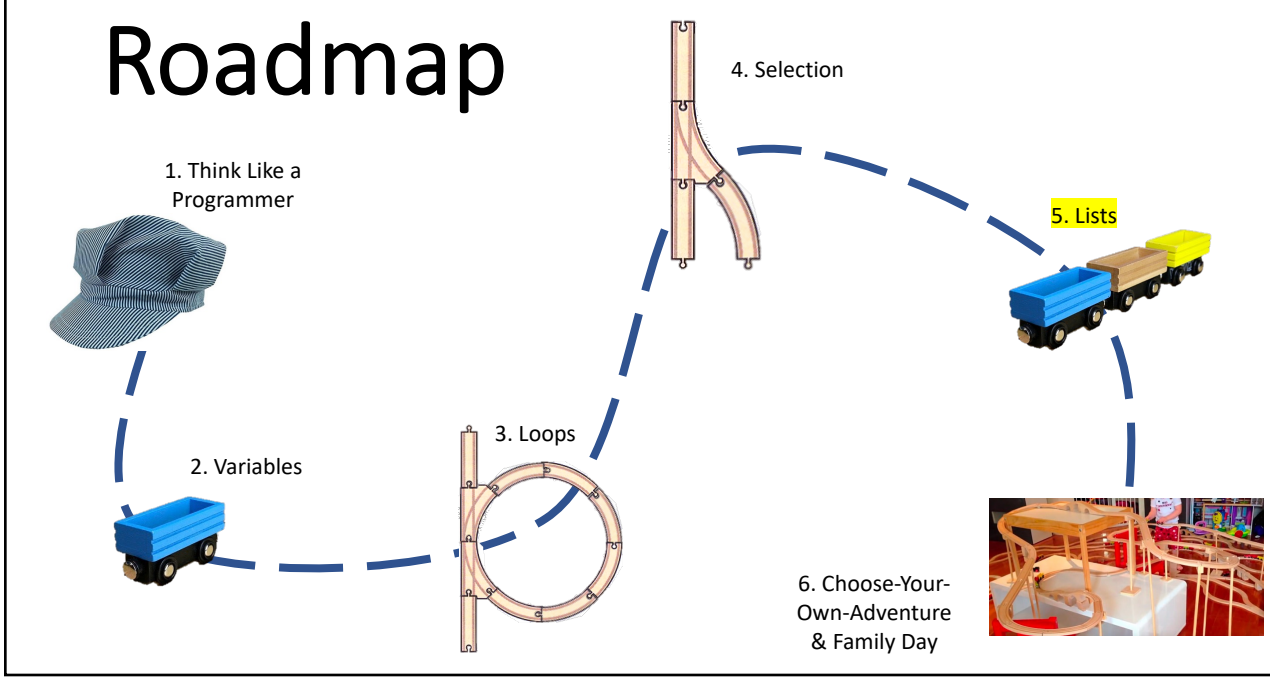
```
II.

if x < 0:
    print("The negative number ",  x, " is not valid here.")
elif x > 0:
    print(x, " is a positive number")
else:
    print(x, " is 0")
```

```
I.

if x < 0:
    print("The negative number ",  x, " is not valid here.")
else x > 0:
    print(x, " is a positive number")
else:
    print(x, " is 0")
```

```
III.

if x < 0:
    print("The negative number ",  x, " is not valid here.")
if x > 0:
    print(x, " is a positive number")
else:
    print(x, " is 0")
```

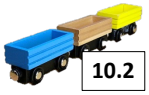
155

Roadmap



156

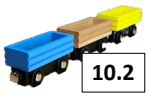
We used *lists* when we did `for` loops




```
print("Hey, friend!")

for day in ["Monday", "Wednesday", "Friday"]:
     place = input("Where should we meet on", day, "?")
     print("I will meet you at", place, "on", day)

print("It's a plan, Stan!")
```

157

The syntax of a list



A ***list*** is _____

and **surrounded** by _____

```
[10, 20, 30, 40]
["spam", "bungee", "swallow"]
```

158

A list can contain a variety of data types

10.2

```
["hello", 2.0, 5, [10, 20]]
```

I am a **sublist**

```
vocabulary = ["iteration", "selection", "control"]
numbers = [17, 123]
empty = []
mixedlist = ["hello", 2.0, 5*2, [10, 20]]

print(numbers)
print(mixedlist)
newlist = [ numbers, vocabulary ]
print(newlist)
```

Try it! ActiveCode 1, section 10.2

159

Use the `len(…)` function to get the length of a list

10.3

```
alist =  ["hello", 2.0, 5, [10, 20]]
print(len(alist))
print(len(['spam!', 1, ['Brie', 'Roquefort', 'Pol le Veq'], [1, 2, 3]]))
```

Do all 3 yellow box activities in section 10.3

160

Live Coding Demo— Write a program that creates a list of your best friends. Save the list in a variable with a good variable name. Print the list and then print the length of the list. Your output should look like this:

10.3

```
My friends are are [‘Jason’, ’Adam’, ‘Matt’]
I have 3 friends
```

161

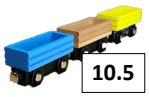
fav_foods.py—Write a program that creates a list of your favorite foods. Save the list in a variable with a good variable name. Print the list and then print the length of the list. Your output should look like this:

10.3

```
My favorite foods are [‘sushi’,’cheese’]
I have 2 favorite foods
```

162

It's easy to check if something is in a list



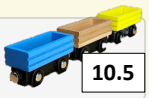

```
fruit = ["apple", "orange", "banana", "cherry"]

print("apple" in fruit)
print("pear" in fruit)

```

163

TAPPS

list-5-1: What is printed by the following statements?

```
alist = [3, 67, "cat", [56, 57, "dog"], [ ], 3.14, False]
print(3.14 in alist)
```



- A. True
- B. False

Check Me | Compare me

list-5-2: What is printed by the following statements?

```
alist = [3, 67, "cat", [56, 57, "dog"], [ ], 3.14, False]
print(57 in alist)
```

- A. True
- B. False

Check Me | Compare me

164

Live Coding Demo—Write a program with a *list* of your closest friends.

10.5

Prompt the user for a name.

Check if the name is in the list.

Print out a message that says whether the name given is one of your friends.

Let the user check five different names.

165

How to **access** just one element of the list

10.4

```
numbers = [17, 123, 87, 34, 66, 8398, 44]
print(numbers[2])
print(numbers[9 - 8])
print(numbers[-2])
print(numbers[len(numbers) - 1])
```

WARNING: In all of CS, indices start at ____

166

TAPPS—What is printed by the following statements?

10.4

```
alist = [3, 67, "cat", [56, 57, "dog"], [ ], 3.14, False]
print(alist[5])
```

A. [ ]
B. 3.14
C. [56, 57, "dog"]
D. “dog”

167

TAPPS—What is printed by the following statements?

10.4

```
alist = [3, 67, "cat", [56, 57, "dog"], [ ], 3.14, False]
print(alist[7])
```

A. ERROR
B. 3.14
C. [56, 57, "dog"]
D. “dog”

168

TAPPS—What is printed by the following statements?

10.4

```
alist = [3, 67, "cat", [56, 57, "dog"], [ ], 3.14, False]
print(alist[3])
```

A. [ ]
B. 3.14
C. [56, 57, "dog"]
D. “dog”

169

Live Coding Demo—Bank Account Balances

10.4

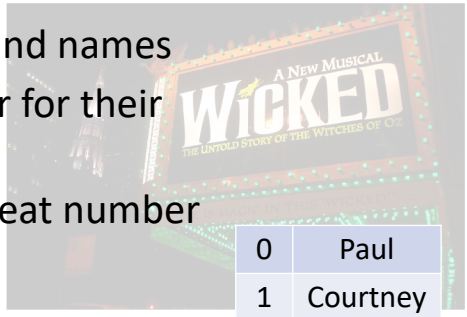
- Pretend that everyone has single-digit bank account numbers
- Write a program that has a list of bank balances (for security make sure some bank account numbers are invalid)
  - `balances = [None, 55.24, None, None, None, 351.90, None, 1900.00, None, 0.00]`
- Prompt the user for a bank account number
- Check if the number is valid (it isn’t `None`)—print an error and exit() if it is invalid
- Otherwise print the account number and account balance
- Let 5 users query their bank account balances

170

seat_list.py—Line order

10.4

- The theater has a list of seat numbers and names
- Create a program that prompts the user for their seat number
- Print out the name of the user at that seat number
- Your program should do this 6 times



0	Paul
1	Courtney
2	Jason
3	Adam

171

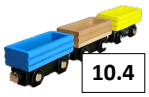
How to **change** an element of the list

10.4

```
numbers = [None, 123, 87, 34, 66, 8398, 45]
print(numbers)
numbers[0] = 17
numbers[-1] = 44
print(numbers)
```

172

TAPPS—What is printed by the following statements?

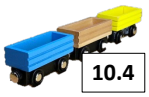



```
alist = [3, 67, "cat", [56, 57, "dog"], [ ], 3.14, False]
alist[2] = 7.65
print(alist[3])
```

A. “cat”
B. 7.65
C. [56, 57, "dog"]
D. 67

173

TAPPS—What is printed by the following statements?




```
alist = [3, 67, "cat", [56, 57, "dog"], [ ], 3.14, False]
alist[2] = 7.65
print(alist[2])
```

A. “cat”
B. 7.65
C. [56, 57, "dog"]
D. 67

174

TAPPS—What is printed by the following statements?

10.4

```
alist = [3, 67, "cat", [56, 57, "dog"], [ ], 3.14, False]
alist[0] = [25, 50, 100]
print(alist[1])
```

A. “cat”
B. 7.65
C. [56, 57, "dog"]
D. 67

175

Live Coding Demo—Bank Account Balances

10.4

- Adapt your bank account program so that after printing the account number and account balance, you prompt the user for a ***new*** balance for that account

176

seat_list.py—Line order

10.4

- Adapt your theater program so that after printing the name and seat number, it prompts the user for a new name to assign to the seat

0	Paul
1	Courtney
2	Jason
3	Adam

177

Accessing lists of lists

10.4

```
food_orders = [[1,9],[10],[2,2],None, None]

order_1 = food_orders[1]
print("Customer 1 ordered", order_1)

for customer_number in range(len(food_orders)):
      print(food_orders[customer_number])

food_orders[3] = [6,1,4,8,3]
food_orders[3][4] = 8
print(food_orders[3][0])
```

How many elements in this list?

Welcome to Burger Queen. Order as many meals as you like! Let me write it down!

178

TAPPS—What is printed by the following statements?

10.4

```
alist = [3, 67, "cat", [56, 57, "dog"], [ ], 3.14, False]
print(alist[3])

A. [ ]
B. ERROR
C. [56, 57, "dog"]
D. “dog”
```

179

TAPPS—What is printed by the following statements?

10.4

```
alist = [3, 67, "cat", [56, 57, "dog"], [ ], 3.14, False]
print(alist[4])

A. [ ]
B. ERROR
C. [56, 57, "dog"]
D. “dog”
```

180

TAPPS—What is printed by the following statements?

10.4

```
alist = [3, 67, "cat", [56, 57, "dog"], [ ], 3.14, False]
print(alist[3][2])
```

A. [ ]
B. ERROR
C. [56, 57, "dog"]
D. “dog”

181

TAPPS—What is printed by the following statements?

10.4

```
alist = [3, 67, "cat", [56, 57, "dog"], [ ], 3.14, False]
print(alist[4][2])
```

A. [ ]
B. ERROR
C. [56, 57, "dog"]
D. “dog”

182

TAPPS—What is printed by the following statements?

10.4

```
alist = [3, 67, "cat", [56, 57, "dog"], [ ], 3.14, False]
print(alist[3][0])
```

A. [ ]
B. ERROR
C. [56, 57, "dog"]
D. “dog”

183

TAPPS—What is printed by the following statements?

10.4

```
alist = [3, 67, "cat", [56, 57, "dog"], [ ], 3.14, False]
print(alist[5][2])
```

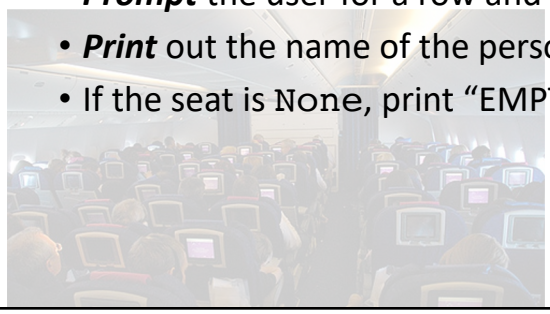
A. [ ]
B. ERROR
C. [56, 57, "dog"]
D. “dog”

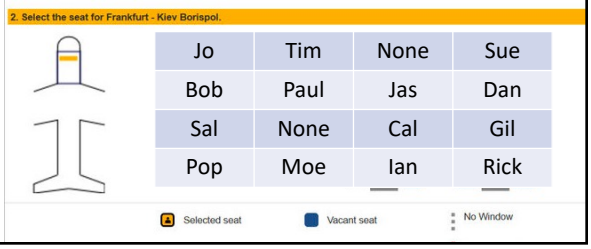
184

seat_list.py—Airline seating chart

10.4

- The flight attendants want a program where they can put in the row and column and find out the passengers name.
- Write a program with a ***list*** of passengers grouped by row
  - `[["Jo","Tim","Kim","Sue"],["Bob",…],…]`
- ***Prompt*** the user for a row and then for a column
- ***Print*** out the name of the person in that row and column
- If the seat is `None`, print "EMPTY"

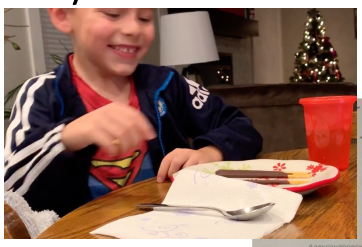




185

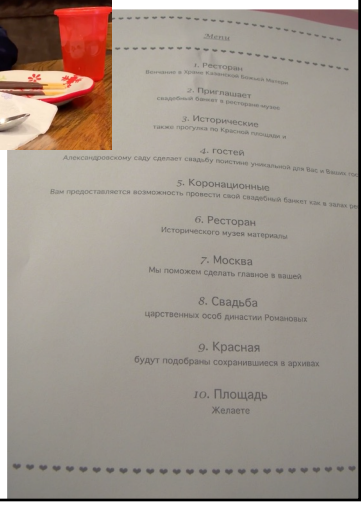
Live Coding Demo—Mystery Dinner!

10.4



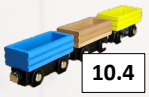
- The dinner has three courses:
  - 0th course: choose 3 dishes
  - 1st course: choose 4 dishes
  - 2nd course: choose 3 dishes
- Example: [[1,5,3], [10,2,6,4], [9,8,7]]
- Write a program with a single variable that contains the choices for each course (you choose, no input)
- Then for each course, print out A) the course number and B) the choices like this:

```
Course 0: 1 5 3
```



186

lockers_list.py—Locker room administrator

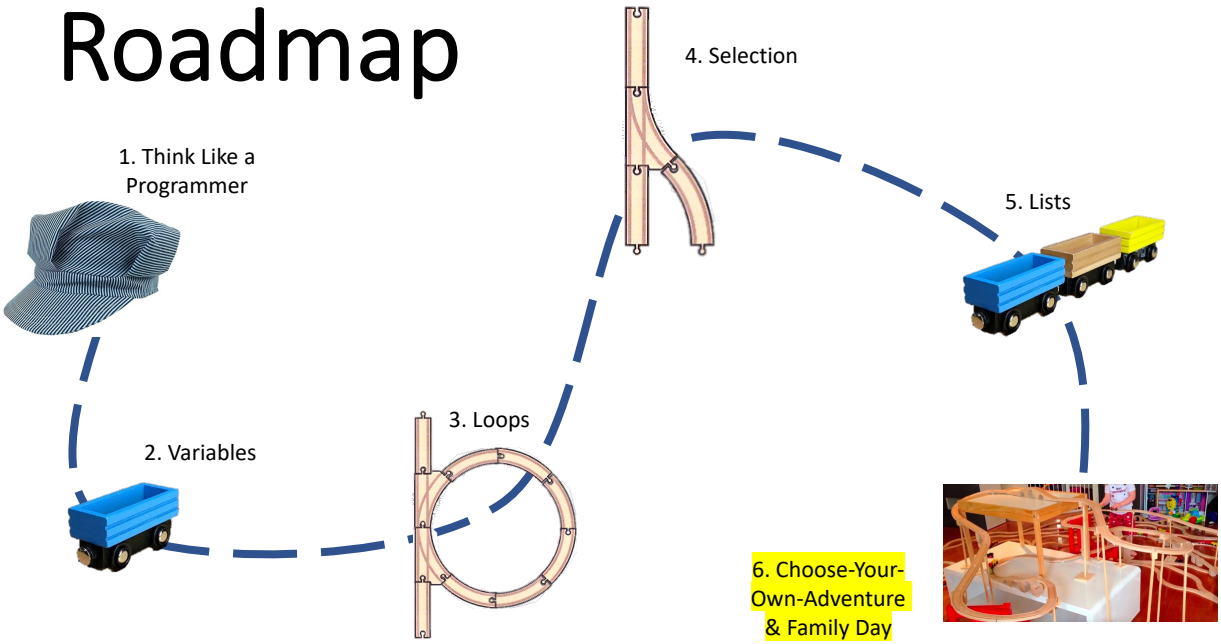


- You are in charge of keeping track of the codes for a set of lockers.
- Write a program that keeps one list with all of the codes (you set the codes)
- Each code should be a ***list*** of three numbers
- Then print out each locker number with its code to look like this:

```
Locker 0 has combo 4 – 6 – 2
```

187

Roadmap



188

Come up with an idea for a quest!

- Setting:
  - Treasure Island (volcano, caves, shipwrecks)
  - Ancient Egypt (pyramid, oasis, mummy's tomb)
  - Amazon Jungle (tree house, river, snake pit)
  - Outer space (bridge, elevator, engine room)
  - Deep Sea Submarine (sea cave, bunk, trench)
  - Cruise Ship (pyramid, oasis, mummy's tomb)
  - School (cafeteria, sports field, band room, office)
  - Haunted House (Kitchen, bedroom, basement)
  - North Pole (science lab, ship, penguin nests)
- Quest:
  - Find a treasure
  - Win a race
  - Destroy a dangerous weapon
  - Turn off a bomb
  - Find a potion to save someone
  - Get food/water
  - Escape a monster/enemy
  - Radio for help
  - Find a lost person who needs help

Keep it simple, pick something with obvious scenes

189

Project: Write a program that prints out the following (or something similar)

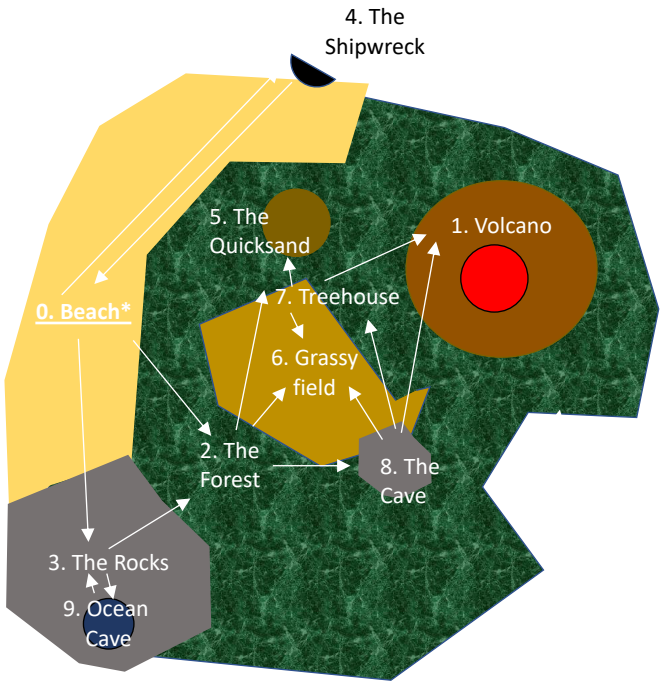
```
Welcome to Treasure Island! Find the treasure. You
have 4  days (moves) before you die!

Day 0 (you have 4 days left)
You are on the beach. You can:
     0. Explore the shipwreck.
     1. Go into the forest.
     2. Explore the rocks.
What do you choose? 1
. . .

After 4 days on the island, you die without finding
the treasure!
```

190

Step 0: Draw a map of the setting

- Pick ~10 scenes that the user can visit on the map. **Number them 0 to 9.** What is the starting scene number?
- Draw paths (**with arrows**) between the scenes. A scene can't have more than 3 choices (out arrows).
- Some of the scenes should be dead-ends (e.g., quicksand, winning place)

Keep it simple, use obvious scenes (like your house)



191

Step 1: Read and understand a program that creates the variables for your adventure.

- It defines variables for:
  - The **number** of scenes in your map
  - The **number** of moves/days
  - A quest **description**
  - A failure **description** for running out of time
  - A **list** of scene descriptions
  - A **list** of choices for each place
- Use comments!
- Don't change variable names or types so we can share adventures!
- Give the file a descriptive name like "treasure_island.py"

What **types**?

Ask me to show you an example!

192

Step 2: Create a separate program called "engine.py" that `import`s your variables

```
import treasure_island as quest
print(quest.quest_description)
```


```
Welcome to Treasure Island! Find the treasure. You have 4  days (moves) before you die!

Day 0 (you have 4 days left)
You are on the beach. You can:
        0. Explore the shipwreck.
        1. Go into the forest.
        2. Explore the rocks.
What do you choose? 1
. . .

After 4 days on the island, you die without finding the treasure!
```

Start with

_____

Code a

_____,

debug as you go,

Keep a

_____

program

193

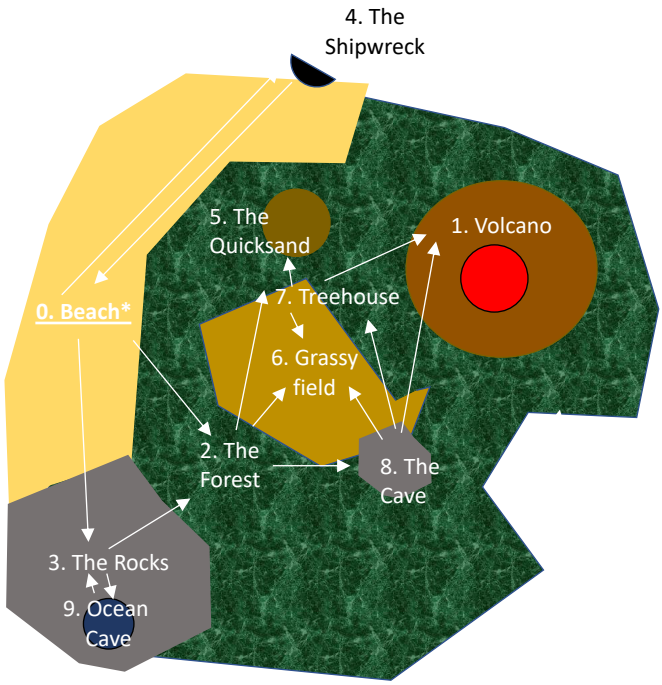
Step 3: Finish writing engine.py

- The program should
  - Print the _____
  - Define the _____
  - Then ***for each*** day:
    - Print out the _____ and _____
    - Print the _____
    - Check if there are _____
      - `exit()` if _____
    - _____
    - Update _____
  - When they run out of days
    - Print the _____

194

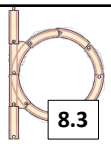
Step 5: Fill in the variable file with places and paths from your quest!

Start ***simple***.

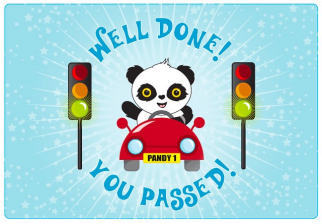


195

A **`while`** loop is like a **`for`** loop, but it keeps looping until an *expression* turns **`False`**




```
score = int(input("What was your score?"))

while(score < 50):
    print("Your score was too low. Try again.")
    score = input("What was your score?")

print("Well done! You passed!")
```



FAMILY DAY

196

number_guessing.py

8.3

Write a number-guessing game!

Pick a hidden number between 1 and 100.
Prompt the user for a guess.
While the user's guess is not the hidden number,
    if their guess is higher, tell them it's higher
    otherwise tell them it's lower.
    Then prompt them for a new guess

Don't forget to tell them when they get it right!

FAMILY DAY

197