Intro to Coding in Python

Lesson 2: Loops and Repetition

Brought to you by the University of Maryland
Balloon Payload Program

Browser based IDE for Python

https://spacepython.com/en/editor/

Find This and Past Lessons:

bpp.umd.edu/wiki

Last Lesson Recap

What we learned last workshop:

- Print Statements
- User Inputs
- Commenting code
- Conditional Statements

Print Statements

Syntax: *print(message)*

Example:

Print strings: *print("Hello World")*
Print numbers: *print(10)*

The print statement **displays** messages to the **console** (area on the right).

User Inputs

Syntax: *input(message)*

Example (try it yourself):

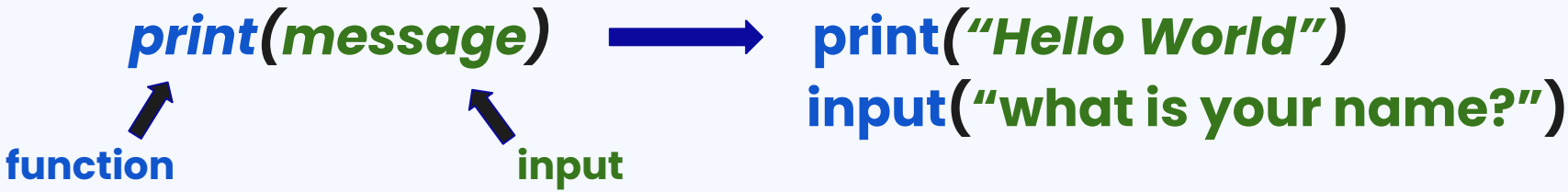
```
1  x = input("what is your name?")
2  print(x)
```

Input is like a print statement, but will wait for the **user** to type in **input** to the **console** before continuing

Calling Functions

Syntax: *function(inputs)*

Example:



Python has many built in functions you can use for convenience (one google search away)

Commenting your Code

Syntax: *#comment*

```
print('hello world') # prints Hello World to the console

# in python anything you write after the '#' is considered
# a comment and will be skipped when the computer executes
# your code

# comment a single line with #

'''
or comment entire portions of code using """ on either side
of the block you want to comment

'''
```

Conditional Statements

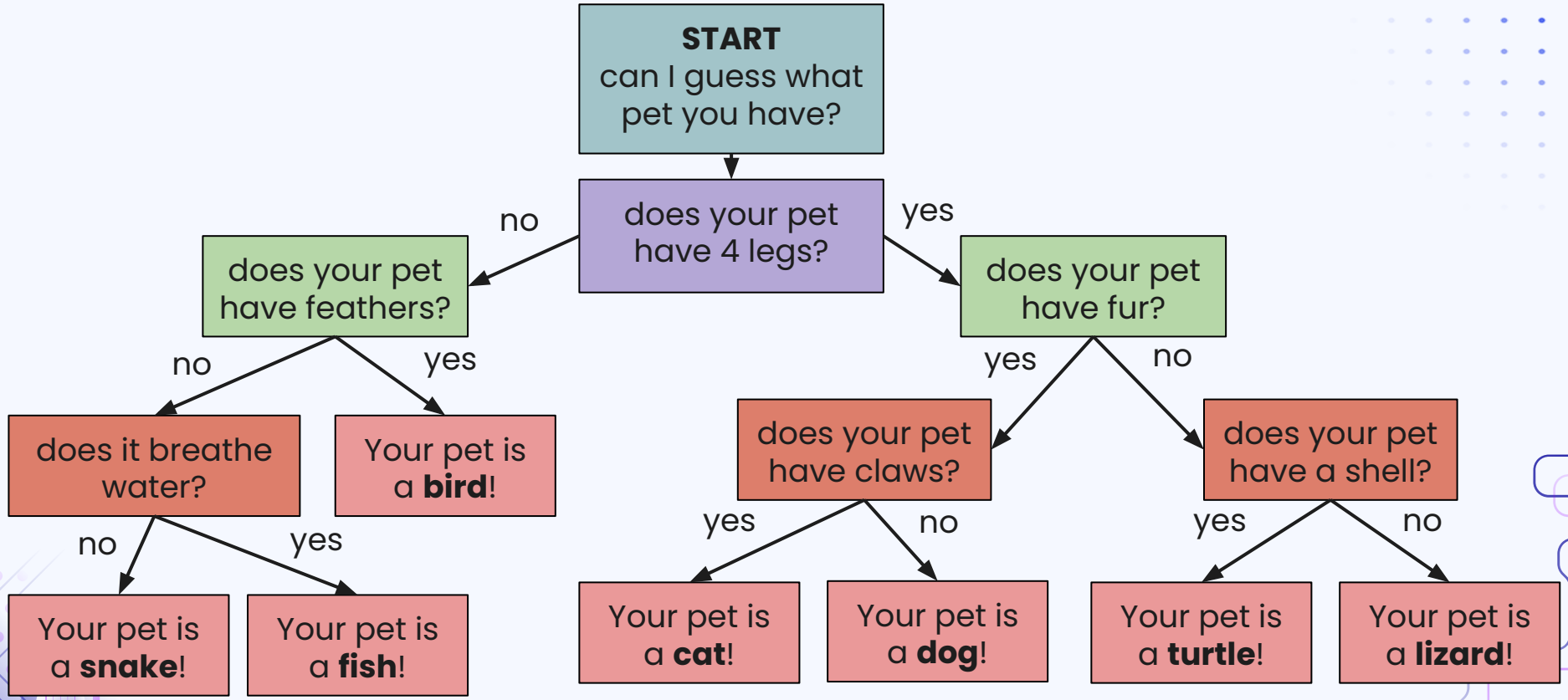
3 different statements: **if, else, else if**

ex)

```
1  x = int(input('give me a random number: '))
2
3  if x == 5:
4      print('x equals the number 5')
5  elif x > 5:
6      print('x is greater than 5')
7  else:
8      print('x is less than 5')
9
10 print('this statement is always executed')
```

practice time

use conditionals to program through this decision tree



Repetition in Code

Python (and coding in general) is great at **automating processes** we do not want to do ourselves. These processes are often **repetitive** and follow a **pattern**.

Good ways to optimize this:

- For loop
- While loop

Today we will learn how to use these loops and different ways to implement them in your code.

What we know so far

Example:

We want our program to count from 1 to 10 by printing each number to the console one by one.

Lets practice:

- can you write a script that prints all all numbers from 1 to 10?

Think:

- **What possible issues could we run into with our current method?**

For Loops

Repeats a block of code for a set number of times

Common syntax:

For i in range():
    Code block

What does this do?

- Initializes i as the counter variable
- Repeats the code block for the number of times inside the function range()

Using the Range Function

Syntax: range(start, stop, step)

- **range()**: Generates a sequence of integers
- **start**: specifies the start number
- **stop**: specifies which number to stop at
- **step**: specifies what number to count by (step size)

Example 1: **range(0, 10, 2)** would iterate through: **0, 2, 4, 6, 8, 10**

Example 2: **range(5, 10)** would iterate through: **5, 6, 7, 8, 9, 10**

Example 3: **range(10)** would iterate through: **0, 1, 2, 3, 4, 5, 6, 7, 8, 9**

Back to our example

We want our program to count from 1 to 10 by printing each number to the console one by one.

program this again, but his time, let's use a **for loop**

Common syntax:

```
For i in range():
    Code block
```

```
number = 1
print(number)

number = number + 1
print(number)

number = number + 1
print(number)

number = number + 1
print(number)

number = number + 1
print(number)

number = number + 1
print(number)

number = number + 1
print(number)

number = number + 1
print(number)

number = number + 1
print(number)

number = number + 1
print(number)
```

vs

```
for i in range(1,11):
    print(i)
```

While Loops

Repeats a block of code until a condition is met

Common syntax:

While condition:

    Code block

Lets try re-writing our example using a while loop

We want our program to count from 1 to 10 by printing each number to the console one by one.

while/for loop pros and cons

For loop

Pros:

- Easier to maintain
- Less potential for error

Cons:

- Must have a pre-specified number of repetitions

While loop

Pros:

- Does not need a preset number of repetitions

Cons:

- Prone to infinite loop errors

Redefining variables inside loops

Notice how we can redefine the same variable using itself inside a loop.

```
i = 0
while i<10:
    i = i + 1
    print(i)
```

There are some nice **syntax shortcuts** we can use for these cases

Addition:

```
i = i + 1
```

```
i += 1
```

Subtraction:

```
i = i - 1
```

```
i -= 1
```

Multiplication:

```
i = i * 1
```

```
i *= 1
```

Division:

```
i = i/1
```

```
i /= 1
```

Typecasting

A common issue you can run into when using input is that python will assign your data the wrong **type.**

How do we **switch** between data types in python?

- **constructor functions**
- Put the variable you are changing the type of inside the ()

String: *str()*
Integer: *int()*
float : *float()*
List: *list()*
etc.

More on Typecasting

Example: Integer to String

```
A = 5

B = str(A)

print(B + B)  % output is “55”
```

Example: String to Integer

```
A = “5”

B = int(A)

print(B + B)    % output is 10
```

Notice how we use operators on different data types

Let's put everything we've learned together

Decide: Program yourself OR Program with me

Using a for or while loop, program the **Number Guessing Game:**

- The user will get 5 chances to guess a number between 1 and 20
- If the guess is too high, the message "too high, guess again" will print to screen
- If the guess is too low, the message "too low, guess again" will print to screen
- If the user guesses the correct number the message "correct, you win the guessing game!" prints to screen
- If the user does not guess correctly within 5 tries, the message "you lost the guessing game" will print to screen