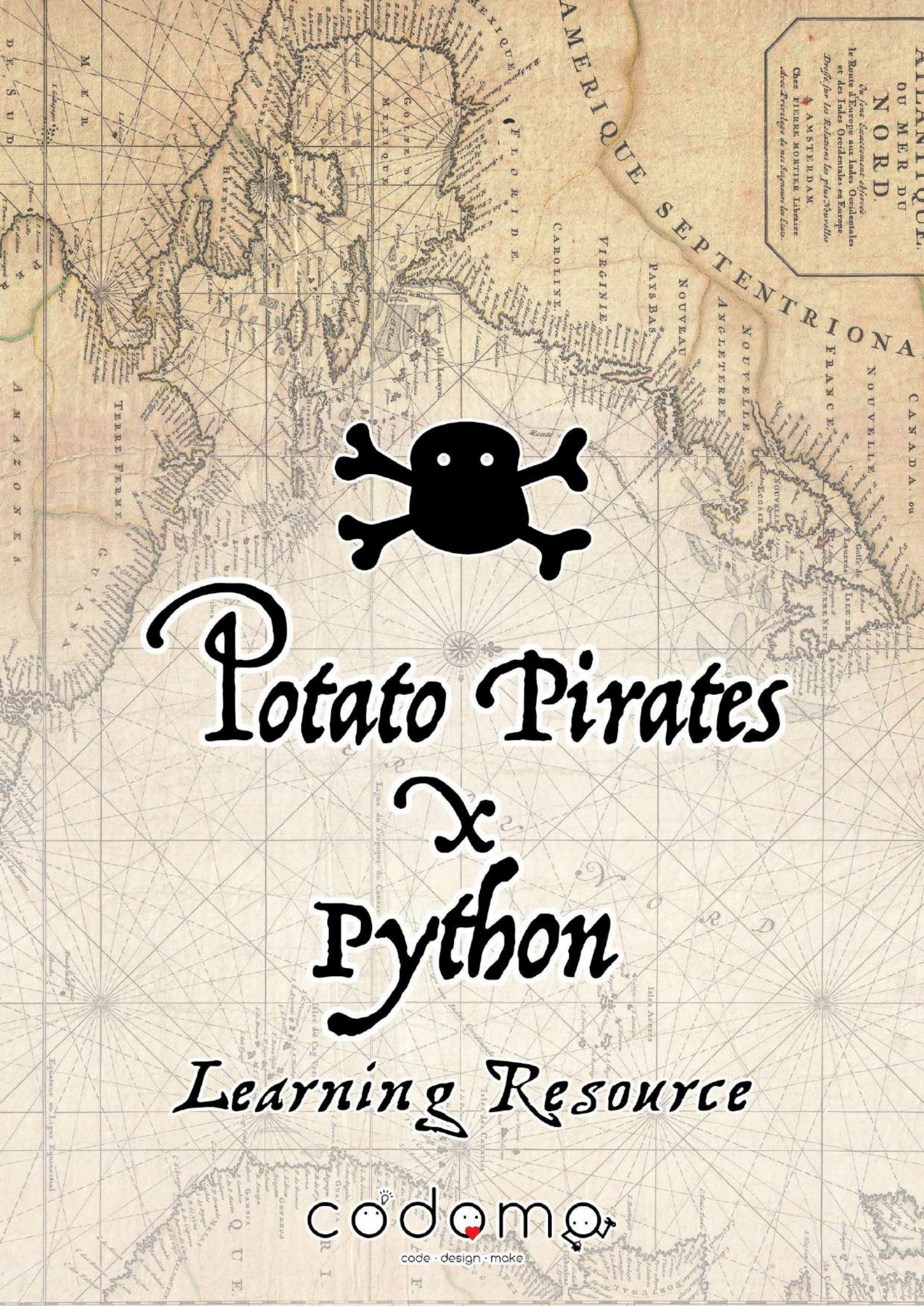


Potato Pirates x Python

Learning Resource

codomo

code · design · make

CONTACT
Website: www.potatopirates.game
Email: ahoy@potatopirates.game, hello@codomo.com.sg

Contents

1. **Getting Started**
   - a. What is Python? 3
   - b. What is a programming language? 3
   - c. Downloading & installing IDLE 4
   - d. My first Python program 5
   - e. Python Shell vs Python Script 5
   - f. How to run my Python program? 6

2. **Chapter 1: Variables**
   - a. What are variables? 9
   - b. Get the value of a variable 10
   - c. Comments 10
   - d. Update the value of a variable 11
   - e. Why do we use variables? 11
   - f. Mathematical operators 12
   - g. Strings 13
   - h. String concatenation 14
   - i. String indexing 15
   - j. Data types 16
   - k. Type casting 16
   - l. Exercise 17

Contents

3. **Chapter 2: For-Loop**
   a. What is a for loop? 20
   b. Breakdown (for loop) 21
   c. Nested for loop 22
   d. Breakdown (nested for loop) 23
   e. Exercise 24

4. **Chapter 3: While Loop**
   a. What is a while loop? 27
   b. Breakdown (while loop) 28
   c. Comparison operators 29
   d. Logical operators 29
   e. Examples 30
   f. Exercise 31

5. **Chapter 4: If-Else and Switch**
   a. What is an if-else statement? 35
   b. Breakdown (if statement) 36
   c. What is a switch statement? 37
   d. Exercise 38

6. **Python Syntax Cheat Sheet** 41

7. **Answer Sheet** 42

Getting Started

What is Python?

Python is a programming language that some of the most popular websites in the world are built with. This includes YouTube, Reddit, Spotify and Pinterest. It has also been used to develop Dropbox and many services, softwares and components of Google.

Web Development

Data Analytics

Robotics

Err.. what is programming language?

Just like humans use languages to speak with one another, computers use programming languages to communicate with other computers. Python is one of the languages commonly used to do this.

Human language

Computer language

Getting Started

Download

1. Go to https://www.python.org/downloads/
2. Download Python 3. *The first number determines the main version. (version 3.6.4 is the latest version at the time of writing).*

Install

1. Run the installation file which has been downloaded.
2. Click "Continue"
3. Accept the software license agreement.
4. You may customise or accept the default installation folder. If unsure, just accept the default settings.
5. Installation should begin. Please wait for a few minutes.
6. Voila! You are now ready to program some Python code.

Opening IDLE

Below are the shortcuts to find IDLE.

Windows

1. Go to Start button.
2. Type "IDLE" in the "*Search programs and files textbox*".

Macintosh OS

1. Press cmd + space
2. Type "IDLE"

IDLE is an **I**ntegrated **D**eve**L**opment **E**nvironment designed to provide us with an environment to edit code, and fix bugs.

Getting Started

My first Python Program

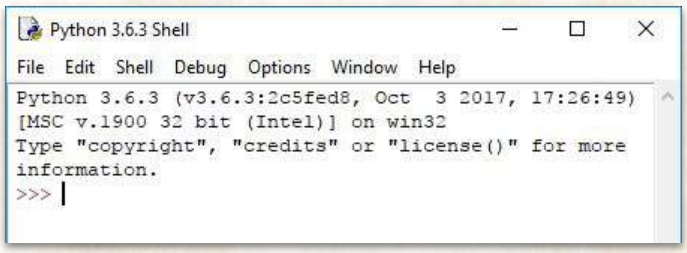
We're going to write our very first Python program that will display "Ahoy Matey" when run.

Ready? Let's go!

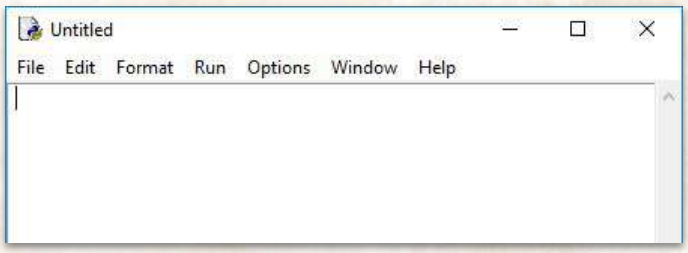
After you have opened IDLE, you will see the window beside.

Python Shell vs Python Script

Python shell is often used to write short codes to check its output. If you are planning to write longer codes, programmers usually use script to do that. Once you run the script, the output will be displayed in the Python shell. To keep things simple, we will write all of our Python code in script, not shell.



Python Shell



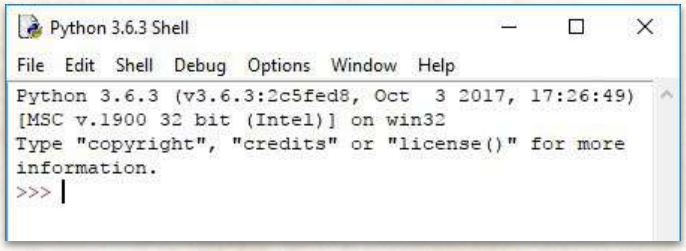
Python Script

Getting Started

How to run my Python program?

1. Opening a Python Script

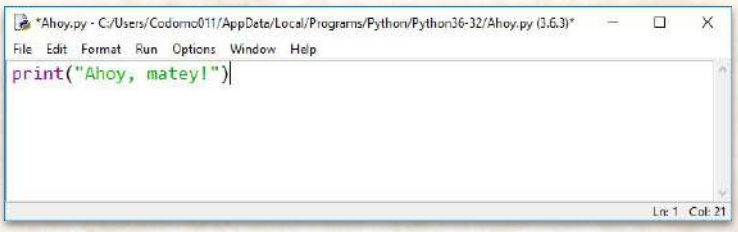
In the Python shell,
Windows: Press Ctrl + N
Mac: Press Cmd + N



2. Writing Code

Type the following code:

```
print ("Ahoy Matey")
```



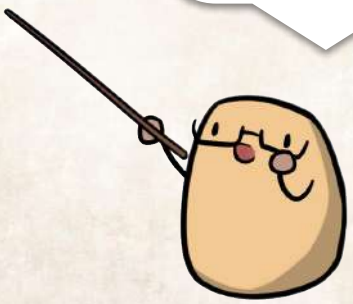
3. Saving your Python Script

Windows: Press *Ctrl + S*.
Mac: Press *Cmd + S*.
Save it in any folder that you like. Just remember where you have saved it and give it a meaningful name because you will create many scripts.

Note: Throughout this guide, you will need to save your python file before running it.

4. Running your Python script

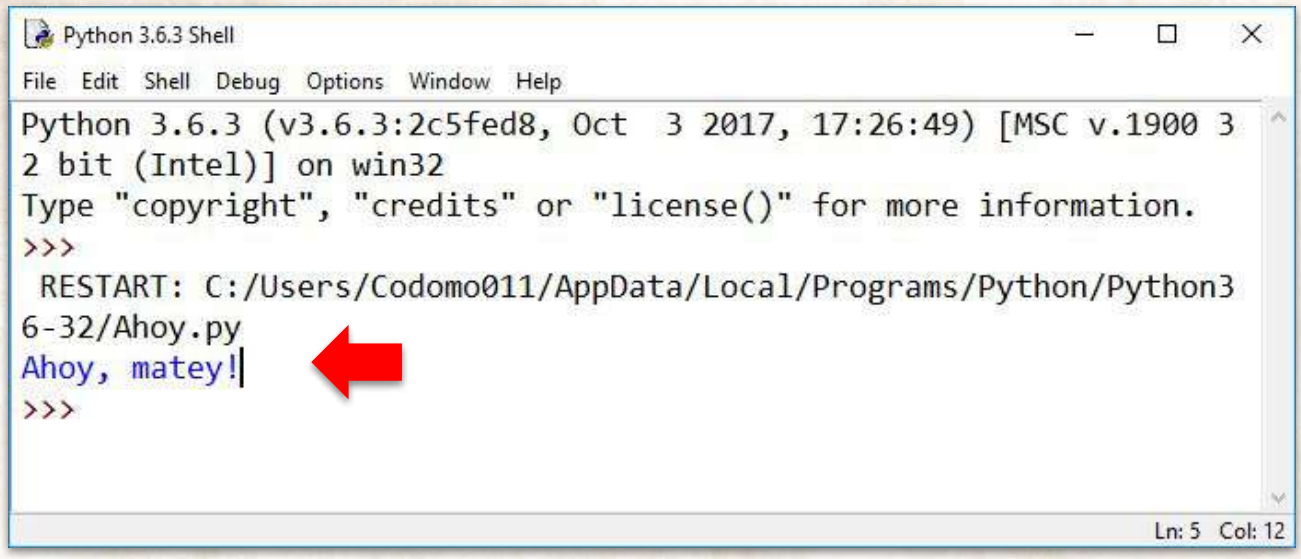
Press F5.
In some laptops, you may need to press the 'fn' key along with the F5 key.

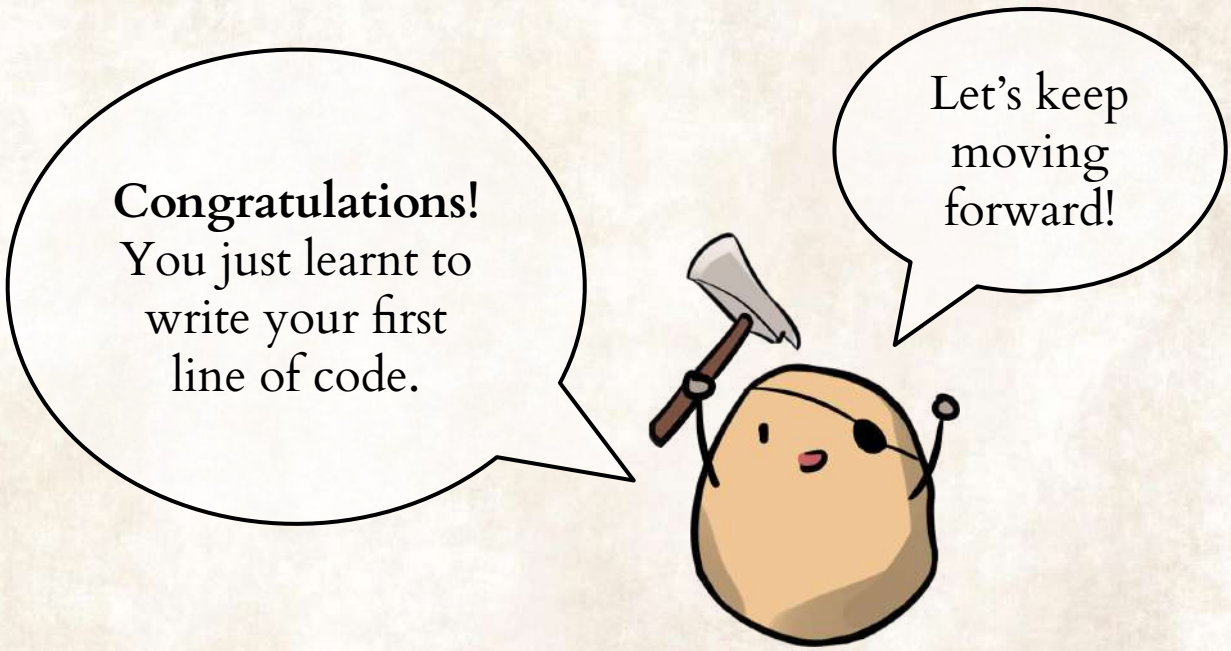


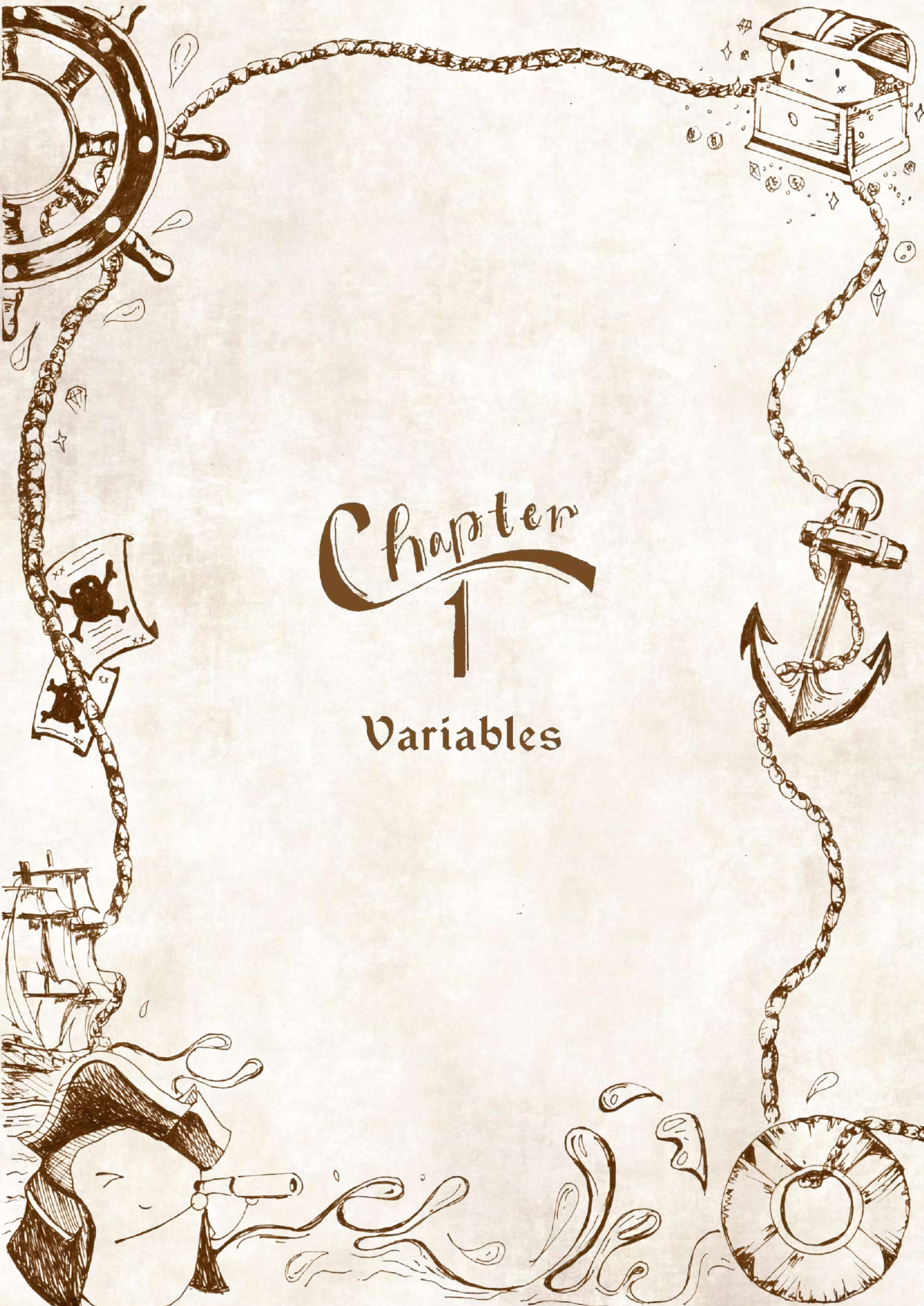
Getting Started

Seeing your result

This window should appear. “Ahoy, matey!” is printed in the shell. As you can see, you can use `print()` to show text. Just remember to use quotes `""` to enclose the text.







Chapter 1

Variables

Variables

What are variables?

A variable acts like a container that stores data. For example, we can use a variable to represent the number of crew members in a ship. Below is a ship with 10 potatoes. Create a variable called "crew" by typing "int crew = 10". "crew" is the variable name, and 10 is the value that it stores. More examples ahead.

In Potato Pirates

In Python

```
crew = 10
```

In Python, we need 2 components to initialise a variable.

Variable name

Every variable is identified with a name. This name can be anything you want as long as it's not a keyword that is reserved in Python for some built-in functions.

```
crew = 10
```

Value

The value stored in this variable "crew" is 10. Later on, you will see that we can store other types of values instead of just numbers.

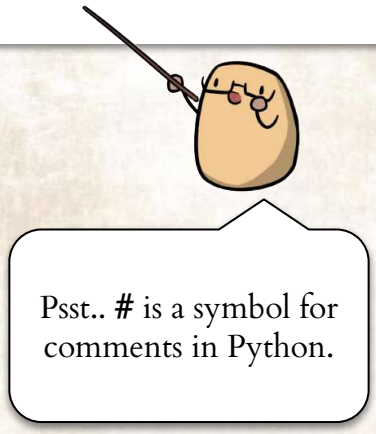
Variables

Get the value of a variable

After storing a value in a variable we can retrieve it by "calling" its name. The code below demonstrates how to retrieve a value stored in a variable.

Python Code:

```
crew = 10 # Set variable "crew" as 10
print(crew) # print the value of "crew"
```



Shell:

```
========== RESTART: C:/Users/s
10
>>>
```

Comments (# symbol)

A comment is a programmer-readable explanation which is usually used to explain the meaning of certain lines of code. Words appearing after # will be ignored by the computer. Very useful!

Variables

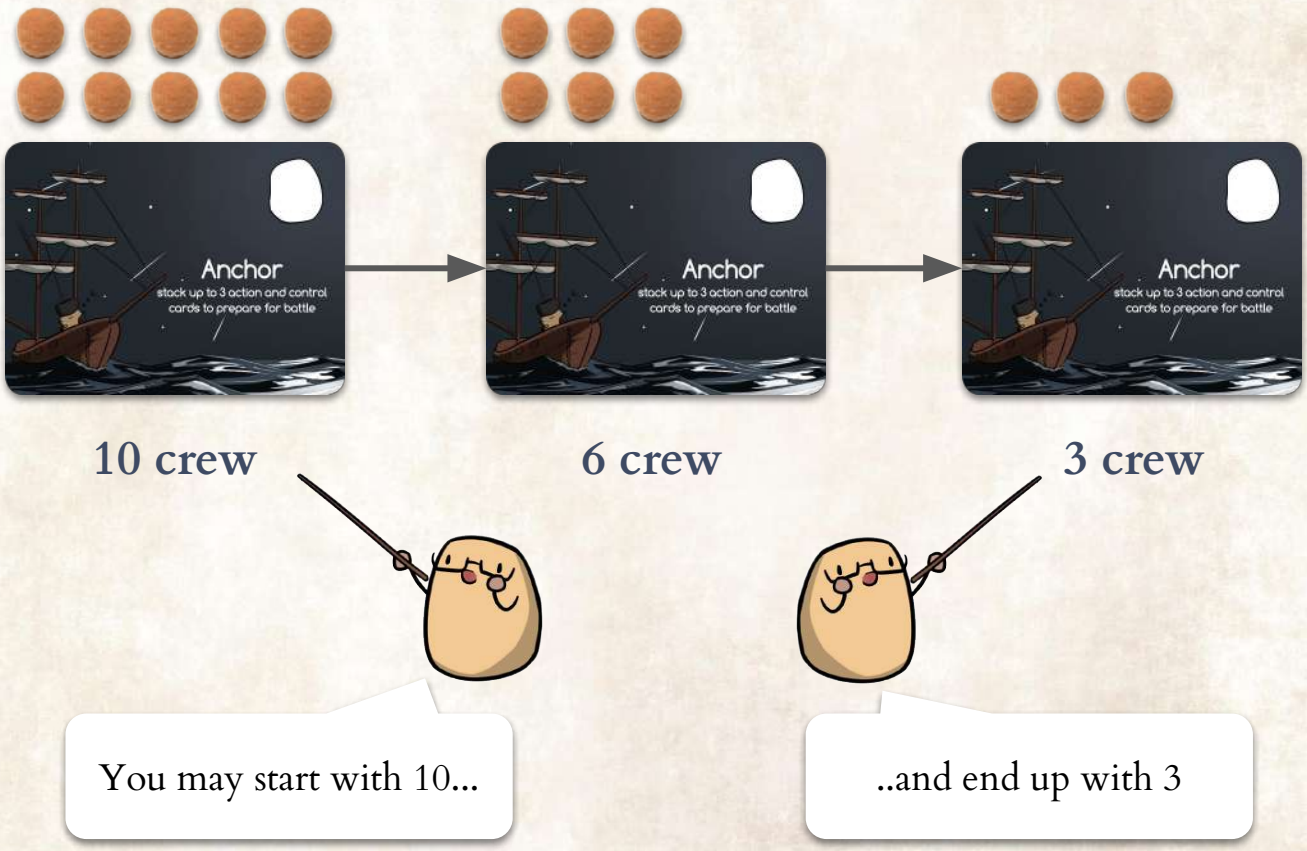
Updating the value of a variable

When you play a Potato King card, you get 2 more potatoes. How do we increase the value of crew by 2? See below.

```
crew = 10 # Set variable "crew" as 10
crew = crew + 2 # Increase the value of "crew" by 2
print(crew) # print the value of "crew"
```

Why do we use variables?

Data is constantly changing. To handle all these changes, the value needs to be stored in an entity, which can then be modified. For example, the number of potatoes on a ship keeps changing as we play the game. We use variables to keep track of all these numbers.



Variables

Mathematical Operators

When writing code, BODMAS order of operations still apply. If you've forgotten your BODMAS rules this could help jog your memory:

Brackets: ( ) Division: / Addition: +

B O D M A S

Order Multiplication: * Subtraction: -

Modulo Operator (%)

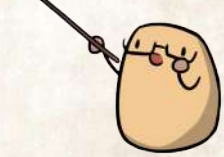
There is also the modulo (or remainder) operator: %. It returns the remainder when the number to its left is divided by the number to its right, for e.g. 9 % 2 = 1

Can you guess the answer?

1) (4+3)-2 My Prediction: Python Returns __
2) (27*9)/3 My Prediction: Python Returns __
3) 27%(3+4) My Prediction: Python Returns __
4) 51/(13%10) My Prediction: Python Returns __

Answer

1) 5
2) 81
3) 6
4) 17



Try your best to answer these questions without coding them

Variables

Strings

The word 'string' is really just a fancy word for a string of letters. To let the computer know that you're using a string, put your letters within quotation marks ("").
Try running the code below on IDLE:

Python Code:

```
control1 = "for 3 times, " # Create variable "control1"
action1 = "roast" # Create variable "action1"
print(control1) # print the value of control1
print(action1) # print the value of action1
```

Console:

```
========== RESTART: C:/Users
for 3 times,
roast
>>>
```

Variables

String concatenation

Concatenation? It is just another fancy word for joining strings together (programmers really love making things sound fancy). To join strings together, we use the '+' operator. See the third line.

Python Code:

```
control1 = "for 3 times, " # Create variable "control1"
action1 = "roast" # Create variable "action1"
finalwords = control1 + action1 # Concatenate control1 and action1
print(finalwords) # print the concatenated string
```

Console:

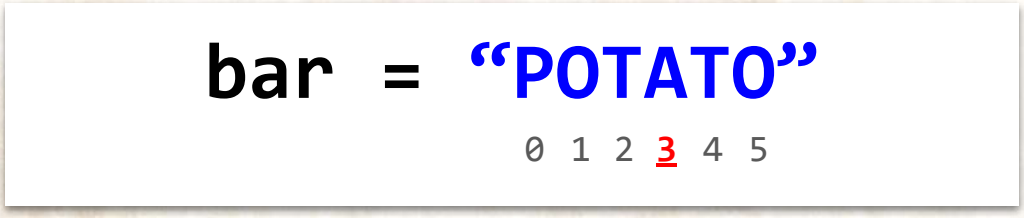
```
========== RESTART: C:/User
for 3 times, roast
>>>
```

Variables

String indexing

Let's say we have a variable named "bar" storing "POTATO" and we want to get the letter 'A' from this string. We first need to figure out the position (i.e. index) of 'A'. If you think the index is 4, you're on the right track, but there's a small catch… The index is 3, not 4. This is because the first index starts from 0, not 1.

Note: foo and bar are just variable names. They don't mean anything.



Let's extract the letter 'A'. Below is the Python code to extract it.

Python Code:

```
bar = "POTATO" # Create variable "bar"
foo = bar[3] # Get the letter 'A'
print(foo) # print the concatenated string
```

Console:

```
========== RESTART: C:/Us
A
>>>
```

Variables

Data types

There are many different data types in programming like integers, strings, double, floats, arrays, tuples, bytes and etc which can be stored as values in variables. We'll only be focusing on integers and string for this learning resource.

Data Type	
Integer	Lists
String	Dictionaries
Character	Tuples
Boolean	*..and many more*

Type casting

Python includes methods to convert from one data type to another. Run the code below on IDLE to convert.

Convert Integer to String

```
x = 3 # Create variable "x" that stores the integer 3
x_str = str(x) # Converts x from integer to string
print(x_str + " potatoes") # prints the concatenated string
# The final result should print "3 potatoes"
```

Convert String to Integer

```
x = "3" # Create variable "x" that stores the string 3
x_int = int(x) # Converts x from string to integer
print(x_int + 2) # prints the integer
# The final result should print 5
```

Variables (Exercise)

Getting & Updating Variable

1. Initialise a variable "x" with value 10. Then, add 5 to it. The final value of x will print 15.
2. Initialise a variable "a" with value 10. Then, divide it by 2. The final value of "a" will print 5.
3. Initialise a variable "b" with value 12. Then, divide it by 3 & save the remainder. The final value of "b" will print 0.

Mathematical Operators (Predict the answer without using a computer)

```
k = 20 + 10 * 20 -20
print(k)
```

k = ___

```
z = 100 % 49
print(z)
```

z = ___

```
x = 5
x = x + 1
print(x)
y = x + 10
y = 2*x + 1
print(y)
```

x = ___

y = ___

Variables (Exercise)

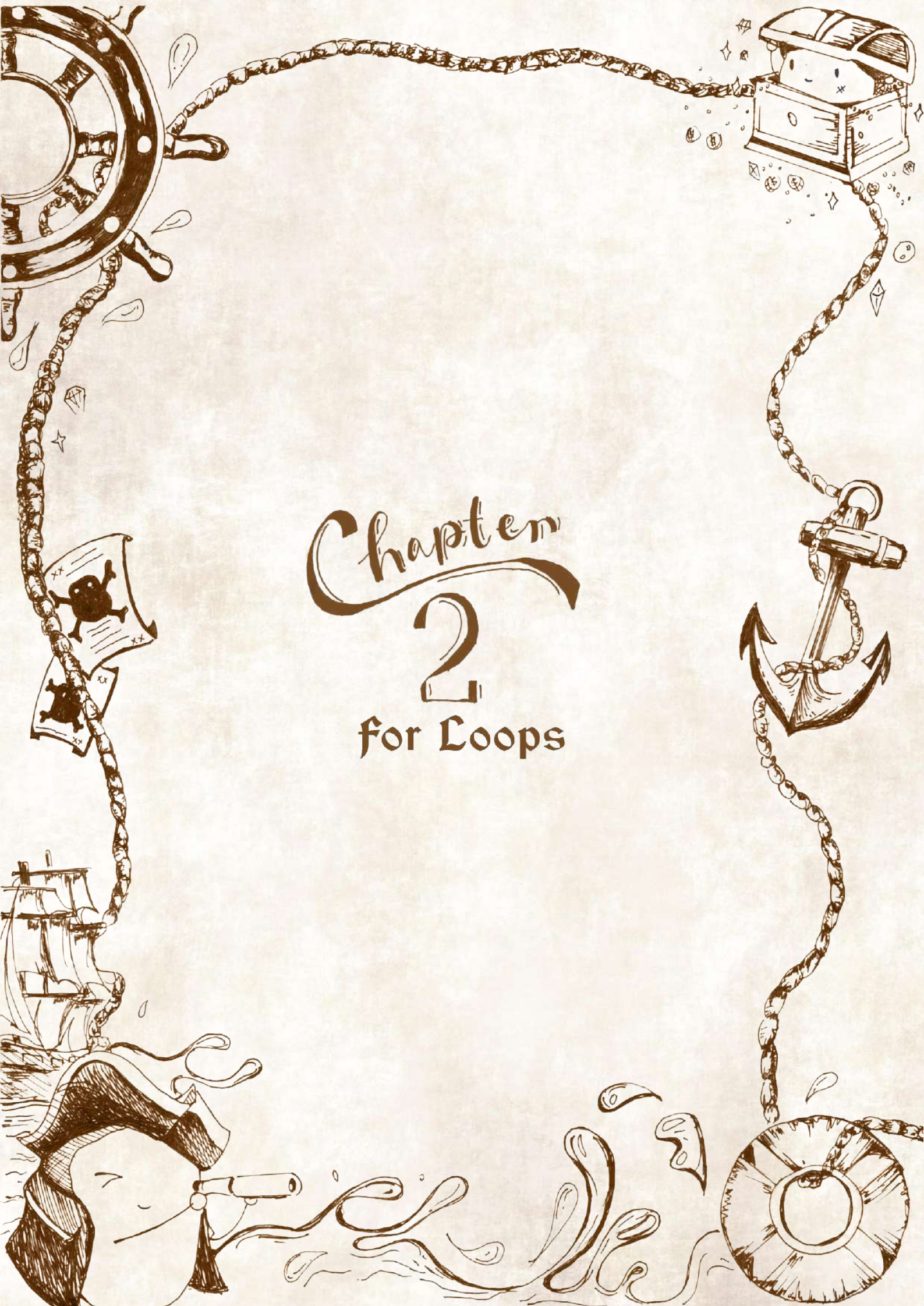
String Concatenation & Indexing

1) a = "Hello, ", b = "Potato King". Concatenate both strings and print out "Hello, Potato King".

2) word1 = "POTATO PIRATES". Get the letter 'S' from word1.

Data Types

1) x = "10". Convert x to an integer, and add 20 into it. You should get 30.
2) y = 7. Convert y into a string and print "7 Potato Kings".



Chapter 2

for Loops

For Loop

What is a for loop?

Suppose you wanted to print the word "Mash" 1000 times. How would you do it? You can write *print "Mash"* 1000 times but that would be a little silly. Thankfully, *for* loops are here to help! *For* loops make repetitive tasks very easy to perform. Let's see how they are used in Python.

In Potato Pirates

In Python

```
for i in range(0,3,1):
    print("Mash")
```

Type this out in IDLE and see what it does. When you're done, let's take a closer look at each part of the control statement.

For Loop

Breakdown (*for* loop)

Let's take a closer look at the structure of a *for* loop.

variable start stop step

```
for i in range(0,1000,1):
```

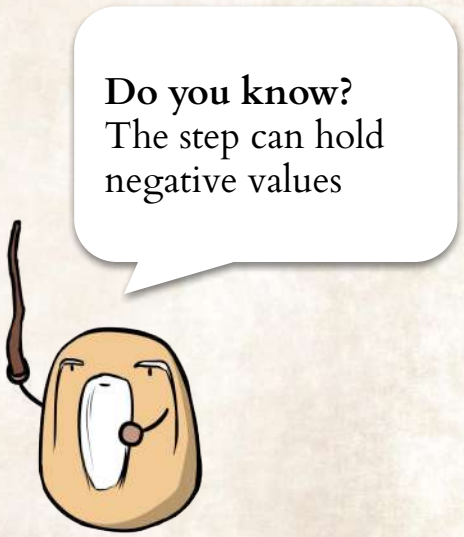
Legend

The letter 'i' is a variable that we use to represent data in the *range* function.
Start: When the loop starts, 'i' will carry this value
Stop: The loop ends if 'i' is more than or equal to this value
Step: Change 'i' by this value from one iteration to the next

Below shows the process of how a *for* loop will work. On the first iteration, the variable "i" will carry the value of 0. On the next iteration, it will increase by 1. The iteration continues until the condition is no longer met.

Iteration	Value of i
1st	0
2nd	1
3rd	2
..	...
1000th	999

Loop terminates when the variable "i" is more than or equal to 1000

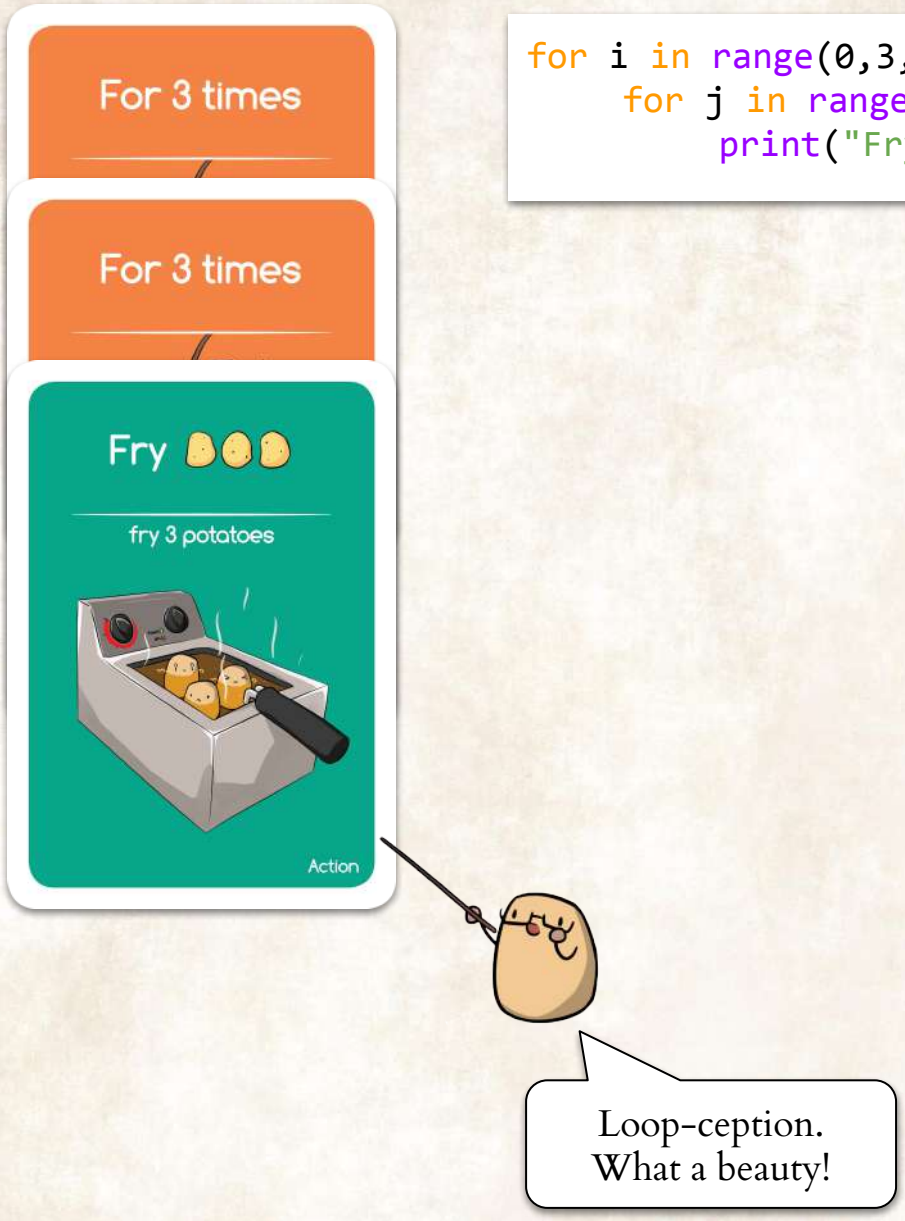


For Loop

Nested for loop

While playing Potato Pirates, did you try stacking several for loop cards on top of each other? You can do the same in Python as well by placing *for* loops into other *for* loops.

In Potato Pirates



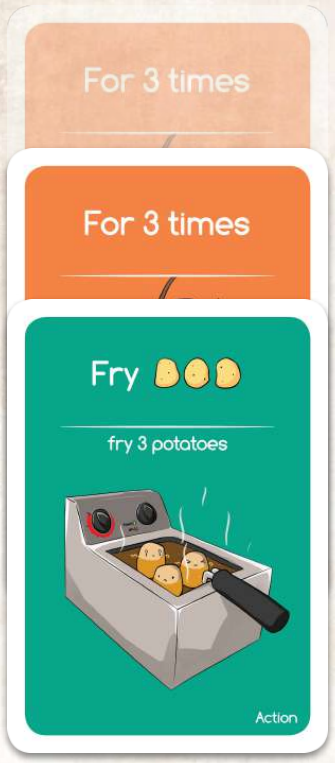
In Python

```
for i in range(0,3,1):
    for j in range(0,3,1):
        print("Fry")
```

For Loop

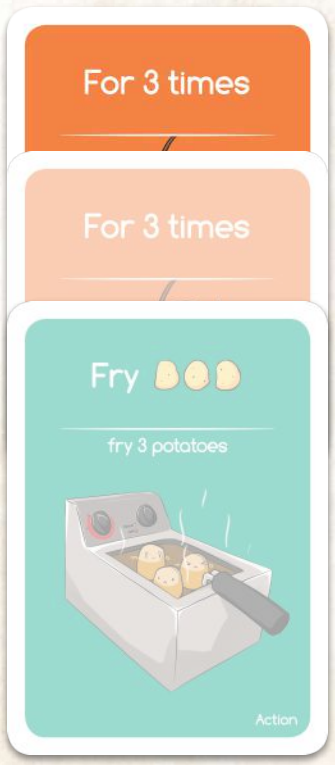
Breakdown (nested *for* loop)

Let's take a closer look



Inner Loop

```
for i in range(0,3,1):
    for j in range(0,3,1):
        print("Fry")
```



Outer Loop

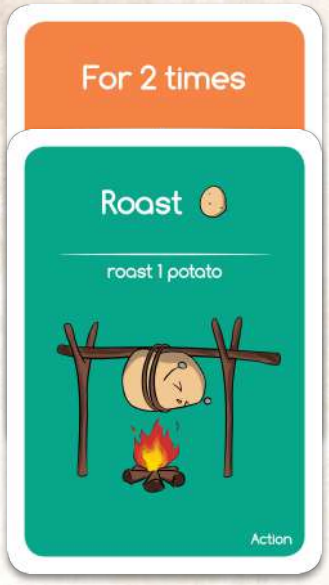
```
for i in range(0,3,1):
    for j in range(0,3,1):
        print("Fry")
```

For Loop (Exercise)

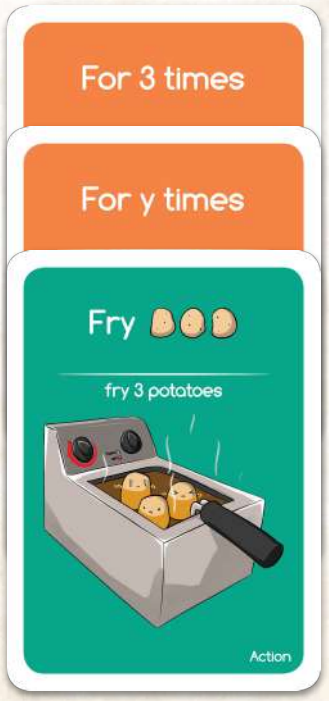
Basic Questions

1) Generate the following sequence of number: 1,2,3,...,100.
2) Generate the following sequence of number: 3,4,..,20.

Convert Potato Pirates to Python



1) An enemy ship has 10 potato crew. You attack it with the cards on the left. Show that the remaining potato crew is 8.

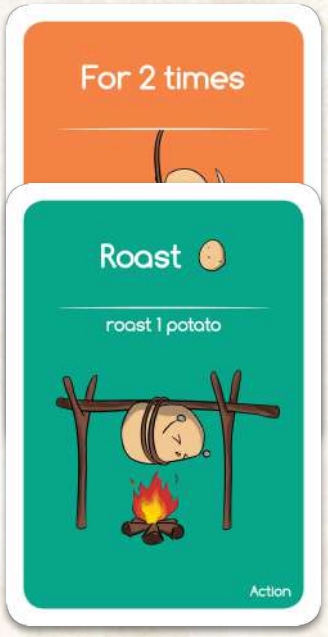


2) The variable y represents the number of ships you have. You have 2 ships, and the enemy has 30 potato crew. Show that the enemy is left with 12 potato crew after the attack on the left.

For Loop (Exercise)

Convert Potato Pirates to Python

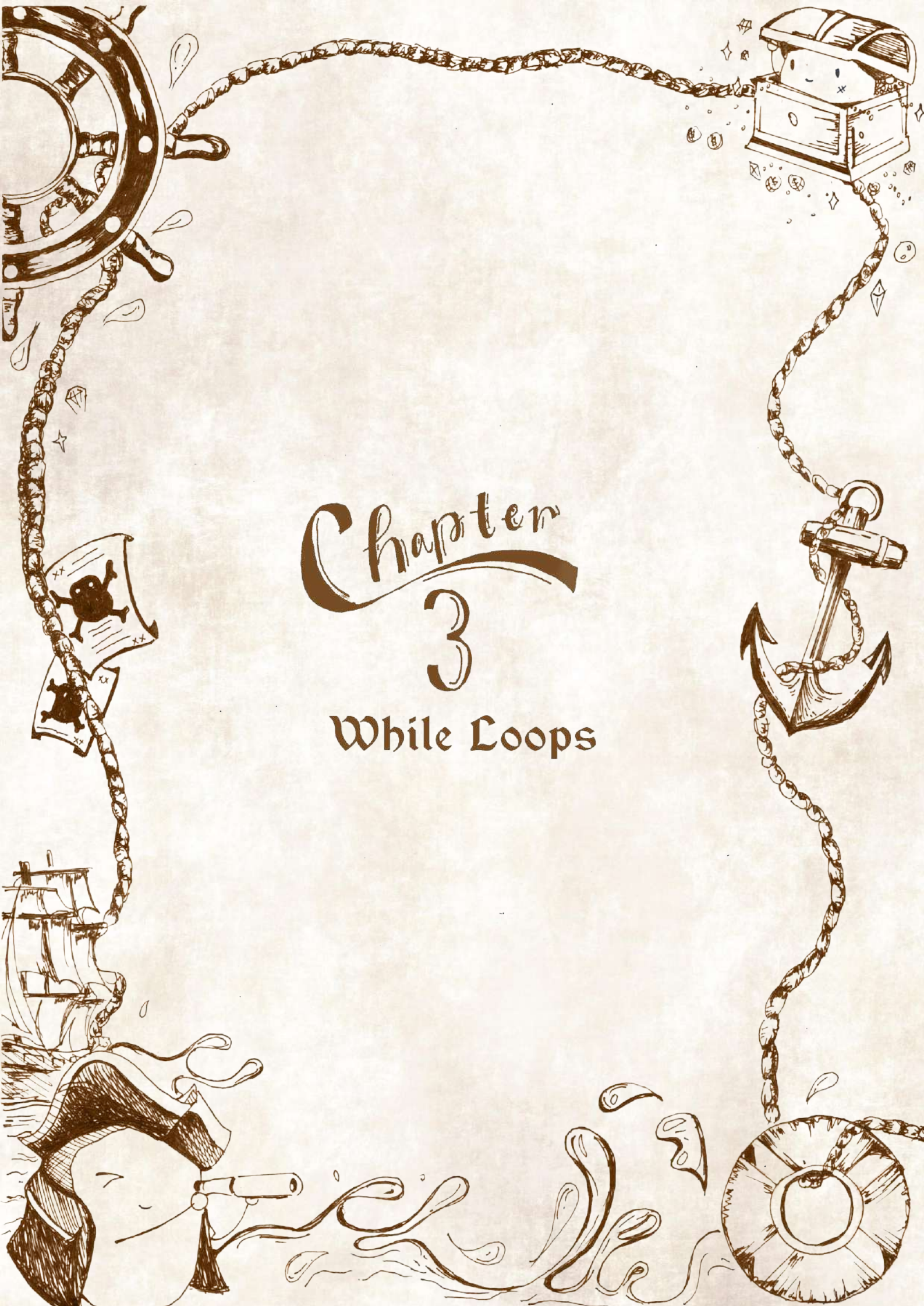
3) An enemy has 10 potato crew members. It was attacked by the card deck on the left, and it has 4 potatoes left. How many cards does the enemy have?



4) Fill in the blank such that the code below will perform the code shown on the cards.

Code:

```
for i in range(5, __, 5):
    enemy_crew = enemy_crew - 1
```



Chapter 3

While Loops

While Loop

What is a while loop?

Unlike *for* loop which repeats based on some numbers, a *while* loop repeats based on a condition. The loop will repeatedly execute its action until the stated condition becomes false. Let's find out how they are used.

In Potato Pirates

In Python

```
while (crew > 4):
    print("Mash")
```

Type this out in IDLE and see what it does. **Remember to declare the variable "crew" first.** You can state crew = 10. When you're done, let's take a closer look at each part of the control statement.

**Note: To force terminate the program, press Ctrl + C*

While Loop

Breakdown (while loop)

Let's take a closer look at the structure of a *while* loop.

condition

```
while (crew > 4):
```

A boolean expression is one which can either take a True or False value. The condition check of "crew > 4" returns a boolean value. The while loop will stop running when the value returned by the condition check is false.

Below shows the process of how a *while* loop will work. Assuming the enemy begins with 7 crew members. On the first iteration, *while* loop will check if the crew is greater than 4. Since 7 > 4 is true, it will execute Mash to reduce the crew by 2. On the next iteration, the Mash action will run again as 5 > 4. Finally, it stops at the third iteration as 3 > 4 is false.

```
crew = 7 # Create a variable with value 7
while (crew > 4) : # While the crew is less than 4
     crew = crew - 2 # Minus 2 from crew

print(crew) # Print the final value of crew
```

Iteration	Value of crew	Is crew > 4?	Action
1st	7	TRUE	Reduce crew by 2
2nd	5	TRUE	Reduce crew by 2
3rd	3	FALSE	NA

While Loop

Comparison Operators

Comparison operators are used to compare 2 objects which either return 'true' or 'false' (also known as booleans). It's very useful.

Comparison Operators

Equal to	==
Not equal	!=
Less than	<
Greater than	>
Less than or equal to	<=
Greater than or equal to	>=

Quick Examples

Expression	Output
1 == 1	TRUE
1 != 1	FALSE
3 < 4	TRUE
1 > 10	FALSE
5 <= 5	TRUE
8 >= 10	FALSE

Logical Operators

Logical operators are typically used with boolean values. When they are, they return a boolean value. Below are some examples.

Logical Operators

AND	and
OR	or
NOT	!

Quick Examples

A	B	A AND B	A OR B
TRUE	TRUE	TRUE	TRUE
TRUE	FALSE	FALSE	TRUE
FALSE	TRUE	FALSE	TRUE
FALSE	FALSE	FALSE	FALSE

While Loop

Examples

Below are some examples of how comparison and logical operators are used in Python.

Python Code:

```
crew1 = 3
crew2 = 5
isDead = False

print(crew1 == 3)
print(crew2 <= 2)
print(crew1 > crew2)
print(crew1 > crew2 and crew2 == 5)
print(crew1 > crew2 or crew2 == 5)
print(not isDead)
```

Console:

```
========== RESTART: C:/Users/sea
True
False
False
False
True
True
>>>
```

While Loop (Exercise)

Basic Questions

1) Generate this sequence of numbers with while loop: 10,9,...,1.
2) Generate this sequence of numbers with while loop: 10,9,..,5.

Predict the result (Do this without a computer)

```
crew = 20
while (crew >= 8):
    crew = crew - 1
print(crew)
```

crew = ____

```
crew = 20
while (crew == 10):
    crew = crew - 1
print(crew)
```

crew = ____

While Loop (Exercise)

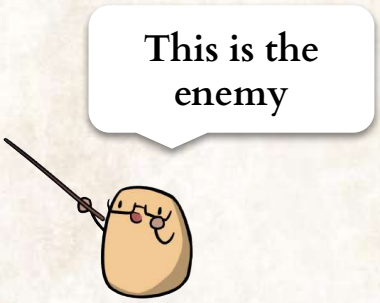
Convert Potato Pirates to Python

1) An enemy ship has 10 potato crew members. You attack it with the card stack shown on the left. Show that the enemy is left with 4 potato crew after the attack.

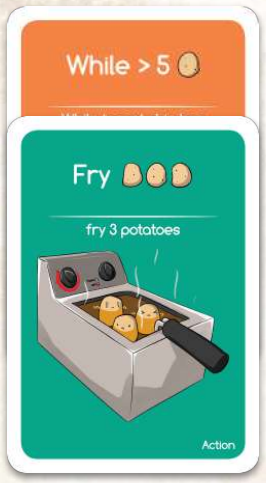
2) An enemy ship has 20 potato crew members. You attack it with the card stack shown on the left. Show that the enemy is left with 2 potato crew.

While Loop (Exercise)

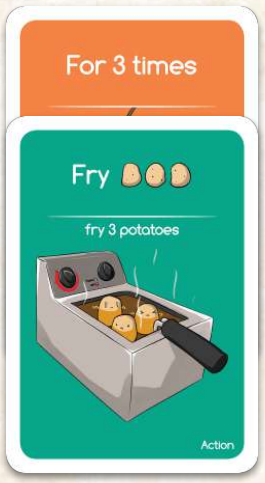
Convert Potato Pirates to Python



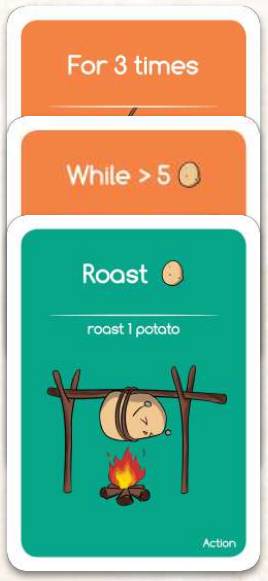
A



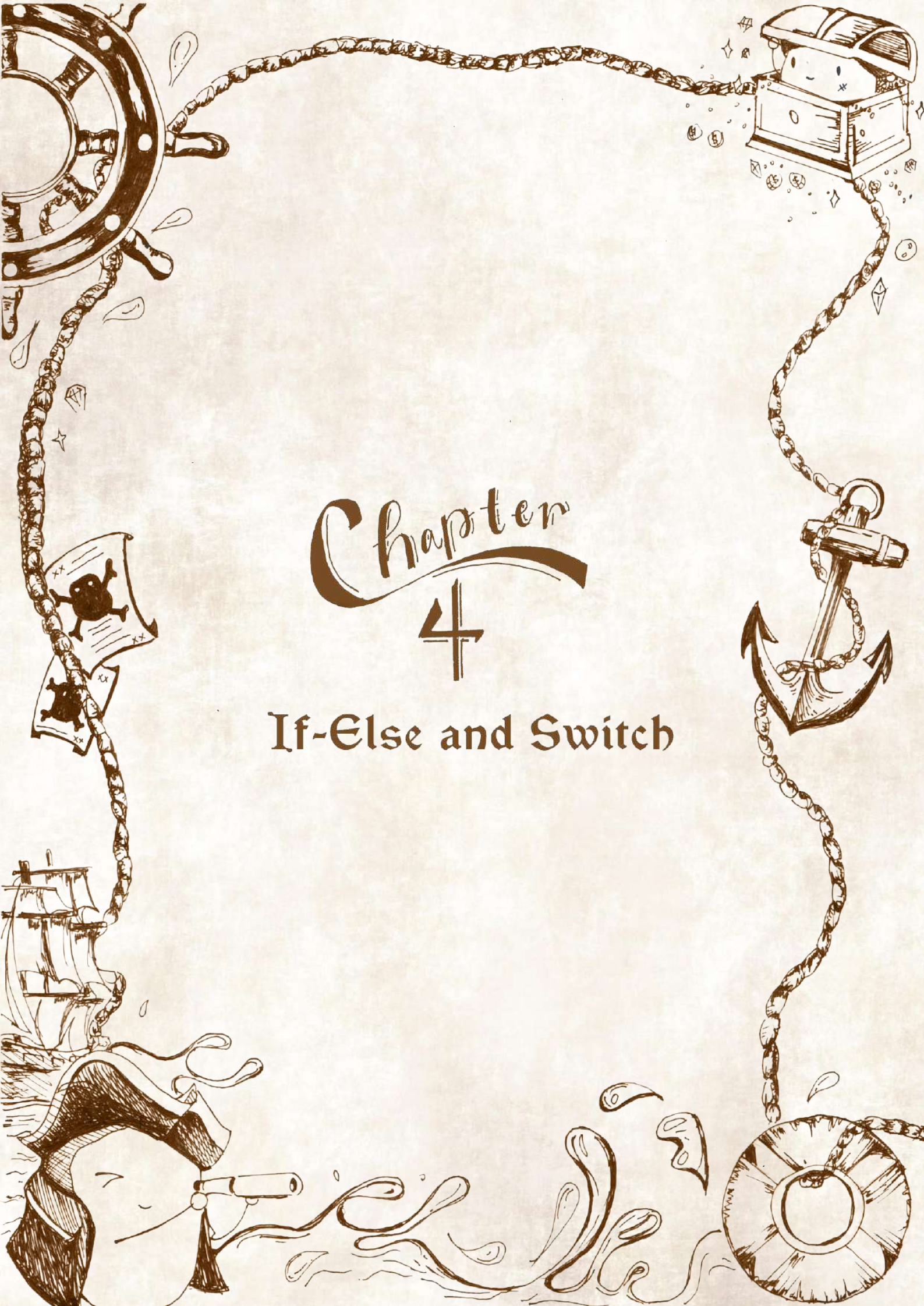
B



C



3) Which card stack can deal the highest damage to the enemy? Code all 3 card stacks in Python.



Chapter 4

If-Else and Switch

If-Else and Switch

What is an if-else statement?
If-else statements help us write programs that can make decisions! We use *if-else* statements everyday in our lives: "If you are tired; go to sleep, or else, continue working", "If you are hungry, eat; or else skip a meal". Can you think of one more?

In Potato Pirates

In Python

```
if (crew <= 4):
     print("Fry")
else:
     print("Mash")
```

Type the code out in IDLE and see what it does. Remember to declare the variable "crew". When you're done, let's take a closer look at the *if* statement.

If-Else and Switch

Breakdown (if statement)

Let's take a closer look at the structure of an if-else statement.

condition

```
if (crew <= 4):
```

```
if (crew <= 4):
    # code A
else:
    # code B
```

Legend

Condition: A boolean expression. It will return True or False

Similar to while loops, *if* statements will execute an action based on a condition. If a condition is True, it will run code A; else it will run code B. It's that simple. However, *if-else* statements differ from *while* loops; they only run once.

If-Else and Switch

What is a switch statement?

Remember switch cards from Potato Pirates? Unfortunately, python doesn't have a *switch* statement. However, we can make use of *if-elif-else* statements to execute similar functions! *If-elif-else* statements allow us to check for additional conditions between *if* and *else*. If the *elif* condition is True, the associated action will be run. Let's try to convert our switch card to python code!

In Potato Pirates


Get a new ship and a potato


Pick a card from the discard pile


Draw 3 cards

In Python

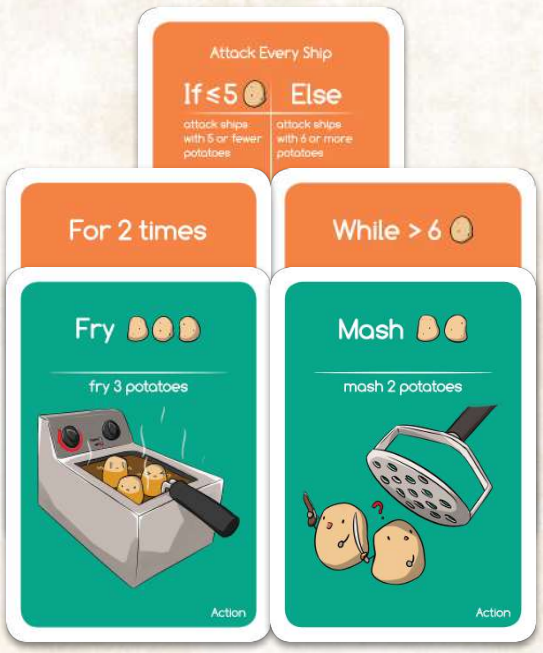
```
crew = 2

if (crew == 1):
     print("get a new ship and a potato")
elif (crew == 2):
     print("pick a card from the discard pile")
elif (crew == 3):
     print("draw 3 cards")
else:
     print("you get nothing")
```

NOTE: The conditions will be checked in order and only the first true statement will be run!

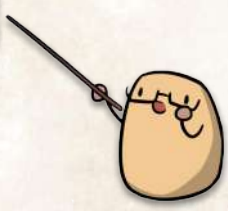
If-Else and Switch (Exercise)

Convert Potato Pirates to Python

1) An enemy ship has 10 potato crew members. You attacked it with the cards shown on the left. Show that the enemy is left with 7 potatoes.



2) An enemy ship has 10 potato crew members. You attacked it with the cards on the left. Show that the enemy is left with 6 potatoes.



*Let's relax the "max 3 cards in your hand" rule for a while

If-Else and Switch (Exercise)

Convert Potato Pirates to Python

3) Create an *if-elif-else* statement with the following terms.
1 Potato : Print("I am dying!")
2 Potatoes: Print("Am I dying?")
3 Potatoes: Print("I am an unlucky pirate")
4 Potatoes: Print("I shall not give up!")
5 or more Potatoes: Print ("There's nothing to worry about")


The End

Python Syntax Cheat Sheet

Initialising Variable

```
crew = 10
name = "PotatoPirates"
```

Update the value of variable

```
crew = 10
crew = crew + 2
print (crew)
```

String Concatenation

```
first = "Potato"
last = "Pirates"
print (first + last)
```

String Indexing

```
name = "PotatoKing"
letterK = name[6]
print (letterK)
```

Type Casting (int to str)

```
var = 10
var_str = str(var)
print (var_str)
```

Type Casting (str to int)

```
var = "10"
var_int = int(var)
print (var_int)
```

For Loop

```
for i in range(0,10,1):
      # action
```

If-elif-else

```
if (crew > 4):
      #action A
elif (crew < 2):
      #action B
else:
      #action C
```

While Loop

```
while (crew > 4):
      # action
```

Comparison Operators

Expression	In Python
Equal	==
Not equal	!=
Less than	<
Greater than	>
Less than or equal	<=
More than or equal	>=

Arithmetic Operators

Expression	In Python
Add	+
Subtract	-
Multiply	*
Divide	/
Remainder	%

Logical Operators

Expression	In Python
And	and
Or	or
Not	not

Answer Sheet for Exercises

Variables

Getting & Updating Variable

1.
```
x = 10
x = x + 5
print(x)
```

2.
```
a = 10
a = a / 2
print(a)
```

3.
```
b = 12
b = b % 3
print(b)
```

Mathematical Operators (Predict the answer without computer)

```
k=200, z=2, x=6 , y=13
```

String Concatenation & Indexing

1.
```
a = "Hello "
b = "Potato King"
print(a+b)
```

2.
```
word1 = "POTATO PIRATES"
s_letter = word1[13]
print(s_letter)
```

Data Types

1.
```
x = "10"
x_int = int(x)
print(x_int + 20)
```

2.
```
y = 7
y_str = str(y)
print(y_str + " Potato Kings")
```

Answer Sheet for Exercises

For loop

Basic Questions

1.
```
for i in range(1,101,1):
      print(i)
```

2.
```
for i in range(3,21,1):
      print(i)
```

Convert Potato Pirates to Python

1.
```
enemy_crew = 10
for i in range(0,2,1):
      enemy_crew = enemy_crew - 1

print(enemy_crew)
```

2.
```
y = 2
enemy_crew = 30
for i in range(0,3,1):
      for j in range(0,y,1):
            enemy_crew = enemy_crew - 3

print(enemy_crew)
```

3. 3
4. 11, 12, 13, 14, or 15

Answer Sheet for Exercises

While loop

Basic Questions

1.
```
crew = 10
while (crew >= 1):
      print(crew)
      crew = crew - 1
```

2.
```
crew = 10
while (crew >= 5):
      print(crew)
      crew = crew - 1
```

Predict the result (Do this without computer)

1. crew = 7 **2.** crew = 20

Convert Potato Pirates to Python

1.
```
enemy_crew = 10
while (enemy_crew > 4):
      enemy_crew = enemy_crew - 2

print(enemy_crew)
```

2.
```
enemy_crew = 20
while (enemy_crew > 4):
      for i in range(0,3,1):
            enemy_crew = enemy_crew - 2

print(enemy_crew)
```

3a.
```
crew = 16
while (crew > 5):
      crew = crew - 3

print(crew)
```

3b.
```
crew = 16
for i in range(0,3,1):
      crew = crew - 3

print(crew)
```

3c.
```
crew = 16
for i in range(0,3,1):
      while (crew > 5):
            crew = crew - 1

print(crew)
```

Card deck A deals the highest damage.

Answer Sheet for Exercises

If-else

Convert Potato Pirates to Python

1.

```
enemy_crew = 10

if (enemy_crew <= 5):
      enemy_crew = enemy_crew - 2
else:
      enemy_crew = enemy_crew - 3

print(enemy_crew)
```

2.

```
enemy_crew = 10

if (enemy_crew <= 5):
      for i in range(0,2,1):
            enemy_crew = enemy_crew - 3

else:
      while (enemy_crew > 6):
            enemy_crew = enemy_crew - 2


print(enemy_crew)
```

Answer Sheet for Exercises

If-else

Convert Potato Pirates to Python

1.

```
crew = 10

if (crew == 1):
      print("I am dying!")
elif (crew == 2):
      print("Am I dying?")
elif (crew == 3):
      print("I am an unlucky pirate")
elif (crew == 4):
      print("I shall not give up!")
else:
      print("There's nothing to worry about")
```

Notes

Notes