Intro to python for text-based adventure games

This document can be used as a resource to help guide you in your journey to learning python. It will overlook some of the key parts of the python programming language, explaining why it is done that way and ways that it could be used with real coding examples inside.

Throughout the notebook, there will be mentioned to the things we learn as we develop our knowledge and there will also be times when it will tell you to check out one of the python files, for you to run.

Well, python is one of the most popular programming languages, thanks to it being easy to use and powerful, making it a great language to kickstart anybody's aspirations of learning to code. It is also popular due to its versatility, since python can be used for all sorts of things from creating games like this course will show you, to developing Artificial Intelligence algorithms, to data analysis.

In [ ]:
```
# This is an example of a comment, which is a piece of text that is not run (not tr
# Programmers use these comments to be able to remind ourselves of things
# Comments are a powerful tool as they allow us to write in the editor, without the
```

Variables

A variable is a container that holds a value. In Python, you can create a variable by simply assigning a value to it. They are a crucial part of programming as they allow us to store data, whether this is input from the user or calculated from a algorithm.

We assign a variblie this like:

In [ ]:
```
# Here we are assigning the variable name to the value Charlie
name = 'Charlie'

#Here we assign the age variable the value of 22
age = 22
```

Data Types

Before moving on we need to understand what the different data types are:

- String
- Int
- Float
- Boolean

In [ ]:
```
# Example of String
name = "Charlie"
```

```
# Example of Int
age = 22

# Example of float
averageTestScore = 0.7

# Example of Boolean
placement = True
```

Be careful when using strings specifically, they can be very complex and powerful in python and are very useful. Sometimes we might want to print out some text next to an integer, for this we need to 'cast' the data type which allows us to quickly convert a int to a string

In [ ]:
```
# Will cause error
print('5 multiplied by 5 is :'+5*5)

#This will work
print('5 multiplied by 5 is:'+str(5*5))
```

Inputting and Outputting

When creating command line programs, such as text-based adventure games, the only way that people can interact with you code is by reading the output and then typing a response, so we need to be able to take a users input and also return outputs

In [ ]:
```
## For outputting code we use the print statement
print('Hello World')

# For inputting data we use the input keyword and set it to a variable
userInput = input('Please input your name:')

print('Hello '+userInput)
```

Control Flow

One of the most importatnt thing you can do with code is change what the user would see depending on a certain condition. An If statement allows you to test a condition and then execute code based on whether or not the condition is True or False.

In addition to the basic if statement, python also provides elif and else statements to test multiple conditions.

The elif statement is short of 'else if' and allows you to test additional conditions if the previous condition is false. The else is a catch all statement which is always at the end of a block and will be executed if all the other conditions will fail.

In [ ]:
```
## This is an example of an if block

grade_percentage = 0.75

if grade_percentage > 0.7:
    print('You got a First in the unit')
elif grade_percentage > 0.6:
    print('You got a "2:1 in the unit')
```

```python
elif grade_percentage > 0.5:
    print('You got a 2:2 in the unit')
elif grade_percentage > 0.4:
    print('You passed the unit')
else:
    print('You failed the unit')
```

Control Logic

Python also provides logical operators such as not, or and and, which allow us to combine conditions together and allow us to make much more complex control statements.

- The NOT operator makes the inverse of whatever is given, e.g. True turns to False
- The AND operator executes the condition if both conditions are True
- The OR operator executes the conditon if one of the given conditions is True

In [ ]:

```python
# Example of AND keyword
x = 5
y = 10
if x > 0 and y > 0:
    print('Both x and y are positive')

# Example of OR keyword
x = 5
y = 10
if x > 0 or y > 10:
    print('One or both of X and Y are positive')


# example of Not keyword
is_raining = False
if not is_raining:
    print('It is not raining')
else:
    print('It is raining')
```

List

In python we can store multiple items of data in a list (sometimes called array). Lists are ordered collectiosn of elements that can be of any data type. You create a list by placing a sequence of data seperated by commas inside square brakets.

You can access elements of a list using indexing. Indexing in python starts with 0, so the first element of the list is index 0.

In [ ]:

```python
# Declare an empty list
empty_list = []
# Declare an empty list with a specific size
my_list = [10]
# Declare and set a list
number_list = [1,2,3,4,5,6,7,8,9]
# Show
first_element = number_list[0]
second_element = number_list[1]

print(number_list)
print('First element is : '+str(first_element))
print('Second element is : '+str(second_element))
```

Loops

Python can use loops to repeatedly execute a block of code. There are two types of loops in Python:

- For loops
- Whie loops

For Loops

A for loop is used to iterate a sequence (like a string or list). There are two main types of for loop, for loops using indexs and then for each loops

For Each Loop:

Each iteration of the loop is stored in a variable, which takes on the value of the next element in the sequence, and the code inside is executed

#### For loop with range

The range keyword

In [ ]:
```
number_list = [1,2,3,4,5,6,7,8,9]

# For Loop
for n in number_list:
    print('This is looping without range' + str(n))

# For Loop with range
for i in range(0,len(number_list)):
    print(number_list[i])
```

While loops

A while loop repeatdely executes a block of code as long as a certain condition is True. each iteration of the loop, the condition is checked. If it is true, the code inside the loop is executed, if false then the loop stops.

In [ ]:
```
# Example of a while loop to take a persons name
name = ''
nameNotFound = True

while nameNotFound:
    userInput = input('Please input your name:')
    if len(userInput) > 0:
        nameNotFound = False
        name = userInput
print('Hello '+name)
```

Dictionaries

A dictionary in python is a collection of {key:value} pairs. Each key must be unique and can be different data types such as string or int. The values can be any data type and don't have to be unique.

```
In [ ]: person = {
            'name': 'John',
            'age': 30,
            'gender': 'male',
            'address': '123 Main St'
        }

        # Accessing information
        name = person['name']
        age = person['age']

        print(f"{name} is {age} years old.")

        # Updating information
        person['address'] = '456 Elm St'

        # Adding new information
        person['phone'] = '555-1234'

        # Removing information
        del person['gender']

        # Iterating over the dictionary
        for key, value in person.items():
            print(f"{key}: {value}")
```

Methods

In python a function (method) is a block of code that performs a specific task and can be reused in the program. Functions can take parameters as input and return a value as output.

We define a method using the def keyword, followed by the name of the method and then (). If we want parameters inside this we place the parameter names inside this variable

```
In [ ]: # def is the keyword to create a method
        # x and y are the parameters that are passed into the method

        def add(x,y):
            return x+y

        result = add(3,4)
        print('The result of 3+4 is: '+str(result))
```

Useful ideas for creating a game

Now we have the basics of python we can move onto using these tools to help us making our game. Below are some coded examples and explanations of how you could approach different things within your game.

Presence Check

When taking input from a user we need to be able to validate the users input to make sure that the input is correct so it won't break the code. One of the main checks we can do for this is the Presence Check. The Presence check, checks that the user has input anything in the first place.

```
In [ ]: # This method is going to return True or False
def isPresent(user_input):
    if user_input == '':
        return False
    else:
        return True
```

Menu

One of the best ways we can work with text based games is used a menu to give the user options when starting the game, we can also use this menu layout when giving the user options, such as letting the user choose what item they want to pick up in a room or where the user wants to go, when provided with different options

```
In [ ]: # Menu runs inside the while loop
while True:
    print('Here is a Menu Title')
    print('1. Option One')
    print('2. Option Two')
    print('3. Option Three')
    print('4. Exit Menu')

    #Put the input here
    user_input = input('Please enter the number option you want: ')

    if isPresent(user_input):
        #The user has entered something so it passes the check
        if user_input == '1':
            print('You have chosen Option One')
            #You could call a method here
        elif user_input == '2':
            print('You have chosen Option Two')
            #You could call a method here - maybe send a user to a room ?
        elif user_input =='3':
            print('You have chosen option Three')
        elif user_input =='4':
            #Using the break will get out the loop and will terminate the loop, enc
            print('Goodbye')
            break
        else:
            #Use this as a catch all if the user didn't input one of the options
            print('You have not entered one of the options')
    else:
        print('**********')
        print('You need to input something')
        print('**********')
```

Inventories

One of the things that you might want to include in your game is an inventory system, which allows the player to store and interact with items, which could help build on game mechanics but could also be used for plot points

In [ ]:
```
# We can create a dictionary an inventory using a dictionary
inventory_dictionary = {'Slot1':'', 'Slot2':'','Slot3':'','Slot4':'','Slot5':''}

# This way lets you slot the items in a specific slot for the user but makes it di

# The other method is using a list
inventory_list = []

# Adding items to them both
inventory_dictionary['Slot1'] = 'Item1'
inventory_list.append('Item1')
```

Puzzles

Apart from story, puzzles are the second biggest element that can make your text-based game entertaining to play and keep the player engaged. There are lots of different ways you can approach puzzles in games.

- One way you could do this is through something like a password, where you give the user subtle hints maybe in another section of the game, forcing the player to explore you game further to progress the story of the game (You want to do this at the right time though, try not make it too difficult)
- You could also give the player more stakes, such as a time limit on the puzzle or an attempt limit before they have to come back and have another go at it.

If you are able to really progress with your game you could also think about putting in a hint system for your puzzle, which could help with user retention as if they are able to ask for a hint at anytime it can make them feel

Riddle

In [ ]:
```
while True:
    print('''I speak without a mouth and hear without ears. I have no body,
          but i come alive with the wind, what am I?''')

    answer = input('> ')
    if answer.lower()=='echo':
        print('Well done, you solved the riddle!')
        break
    else:
        print('Sorry try again.')
```

Keys and locks

In [ ]:
```
inventory = []

## Pick up the key somewhere in the game

inventory.append('key')
```

```python
if 'key' in inventory:
    print('You pull the key out your pocket and open the door, YOU ARE FREE')
else:
    print('Seems that I need a key to open then door')
```

Code based

In [ ]:
```python
code = ['2','6','1','8']

while True:
    print('The door is locked and needs a 4-digit code')
    user_code = input('> ')
    if len(user_code)==4:
        digits_correct = [False,False,False,False]
        for i in range(0,4):
            if code[i] == user_code[i]:
                digits_correct[i] = True

        if False in digits_correct:
            if True in digits_correct:
                for i in range(0,4):
                    if digits_correct[i]:
                        print(f'You got posistion {i+1} correct with {user_code[i]
                print('But the rest were wrong')
            else:
                print('You have got one or more digit incorrect')
        else:
            print('You got the code!')
            break
    else:
        print('It must be a 4 digit code')
```