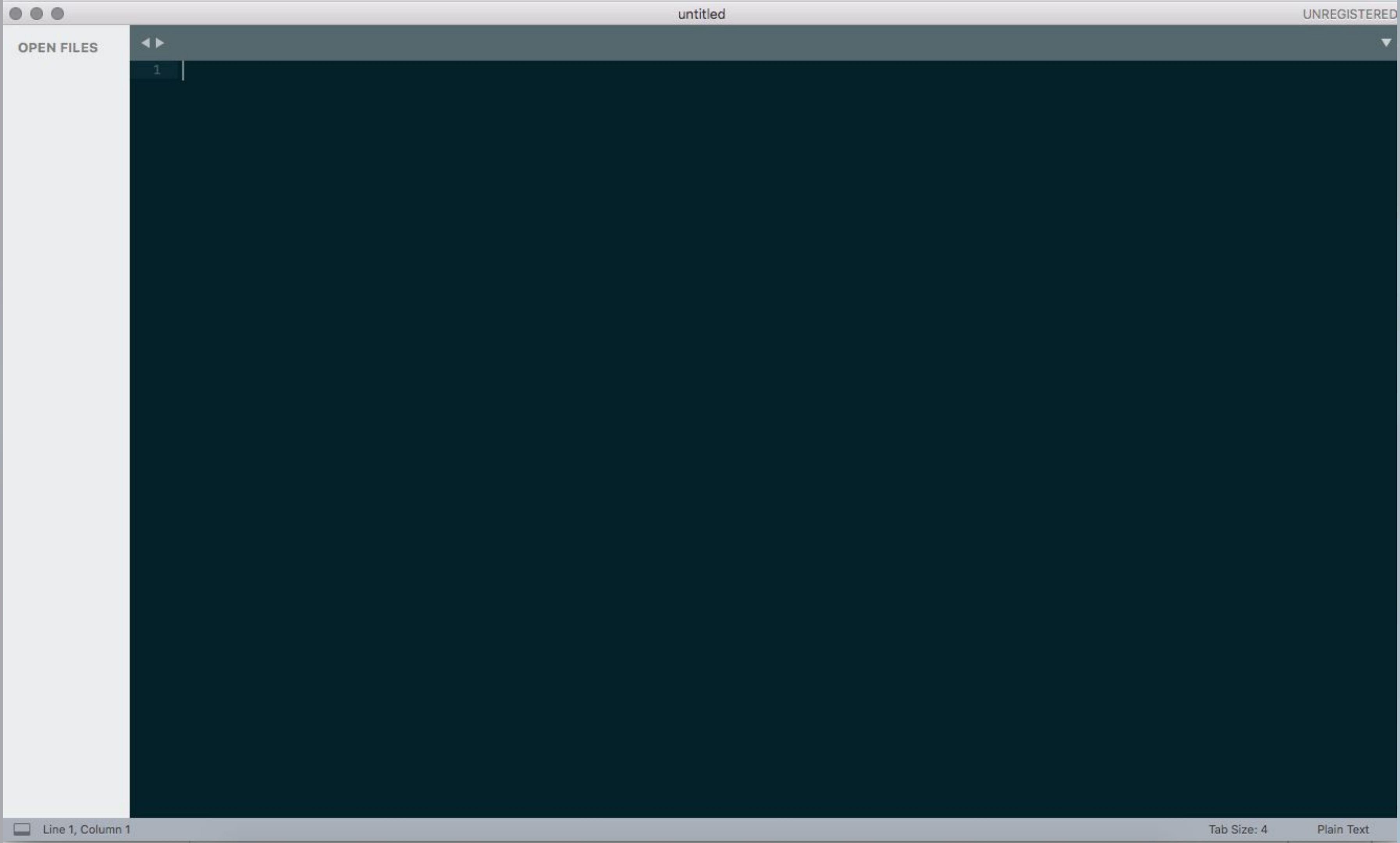
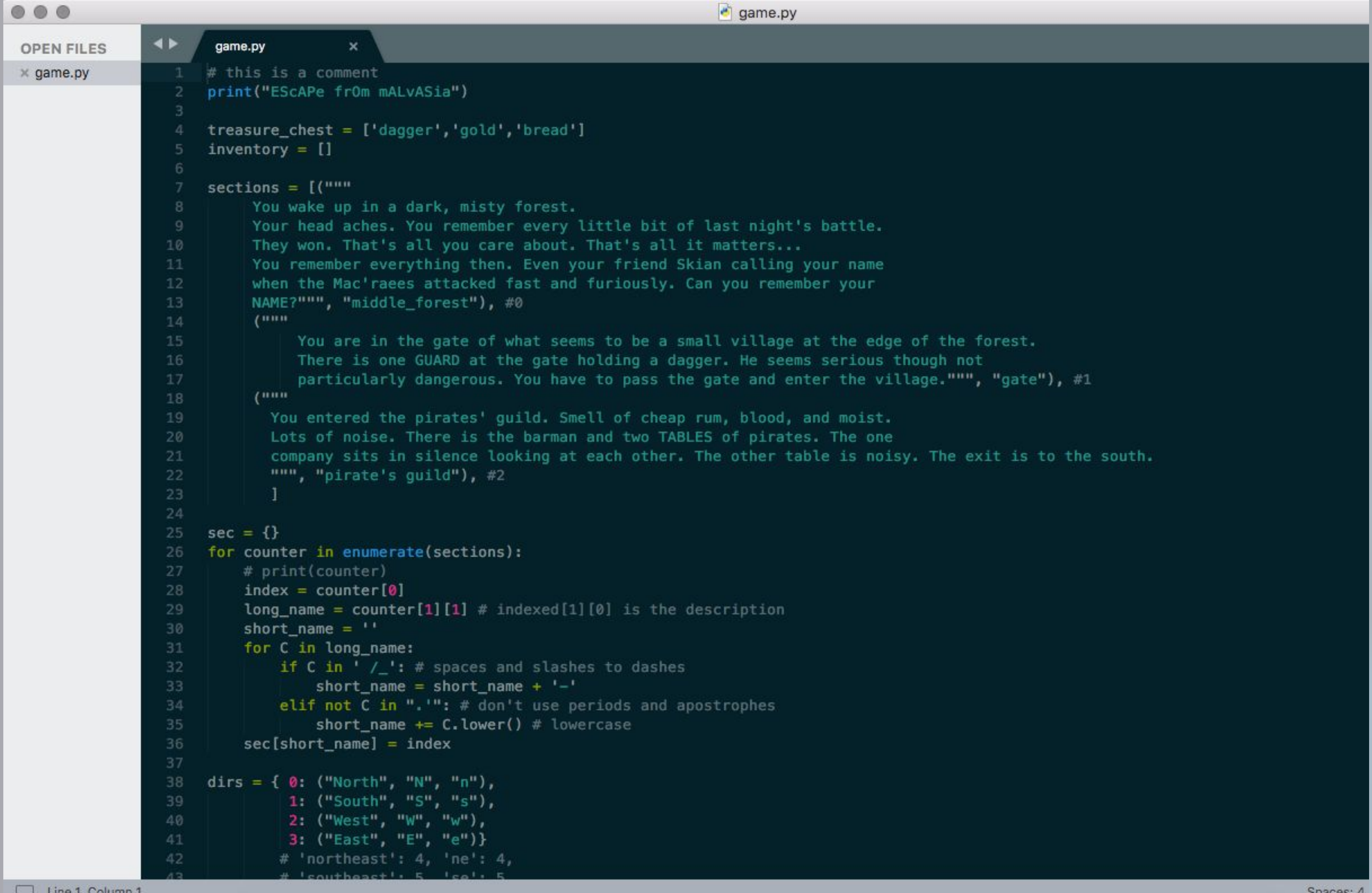
GAMES PROGRAMMING I

Mariza Dima [2018/19]

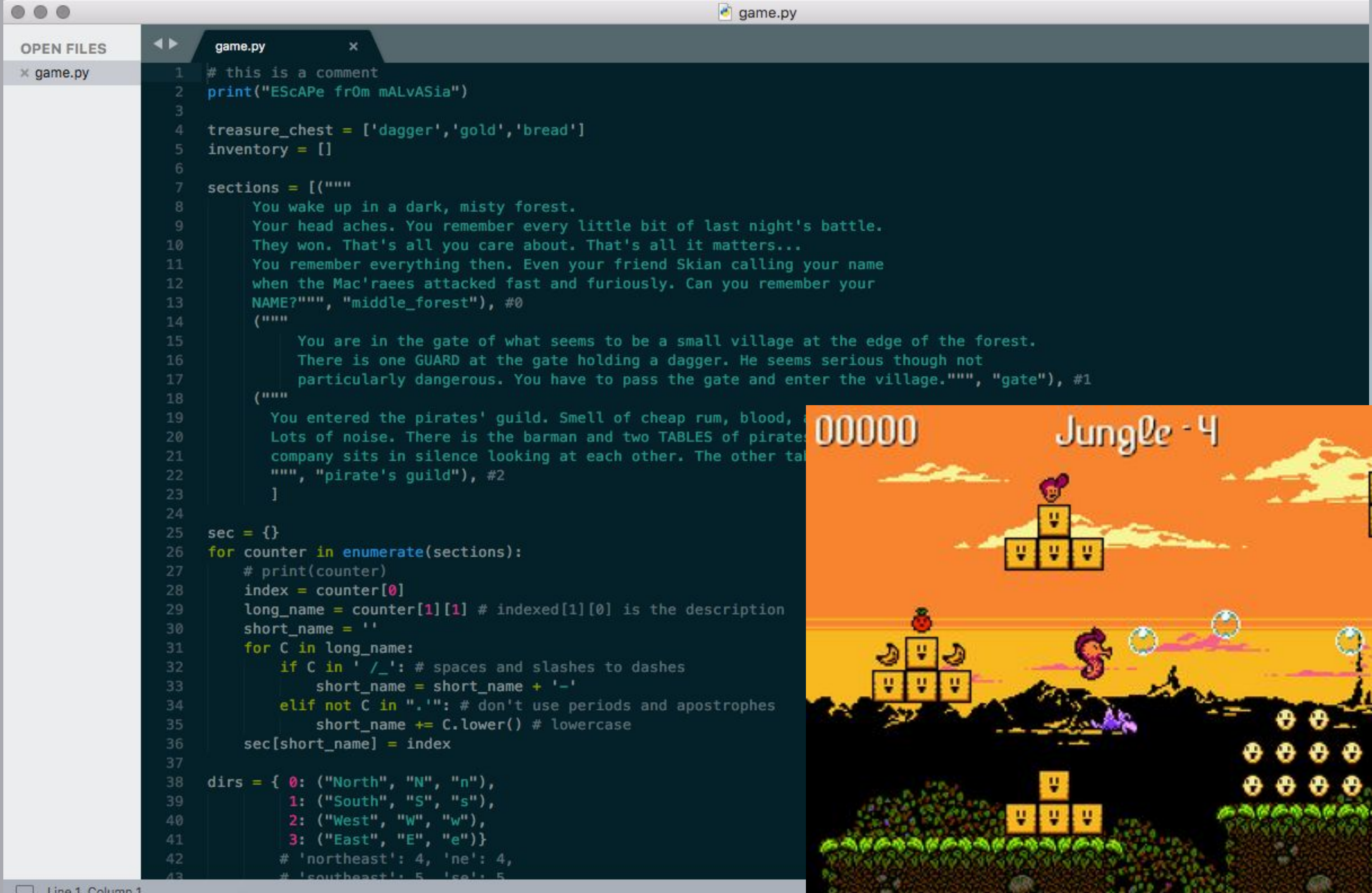
TODAY

- Module and Assessments
- Introduction to Python
- Working with Github


untitled
UNREGISTERED
OPEN FILES
Line 1, Column 1
Tab Size: 4
Plain Text

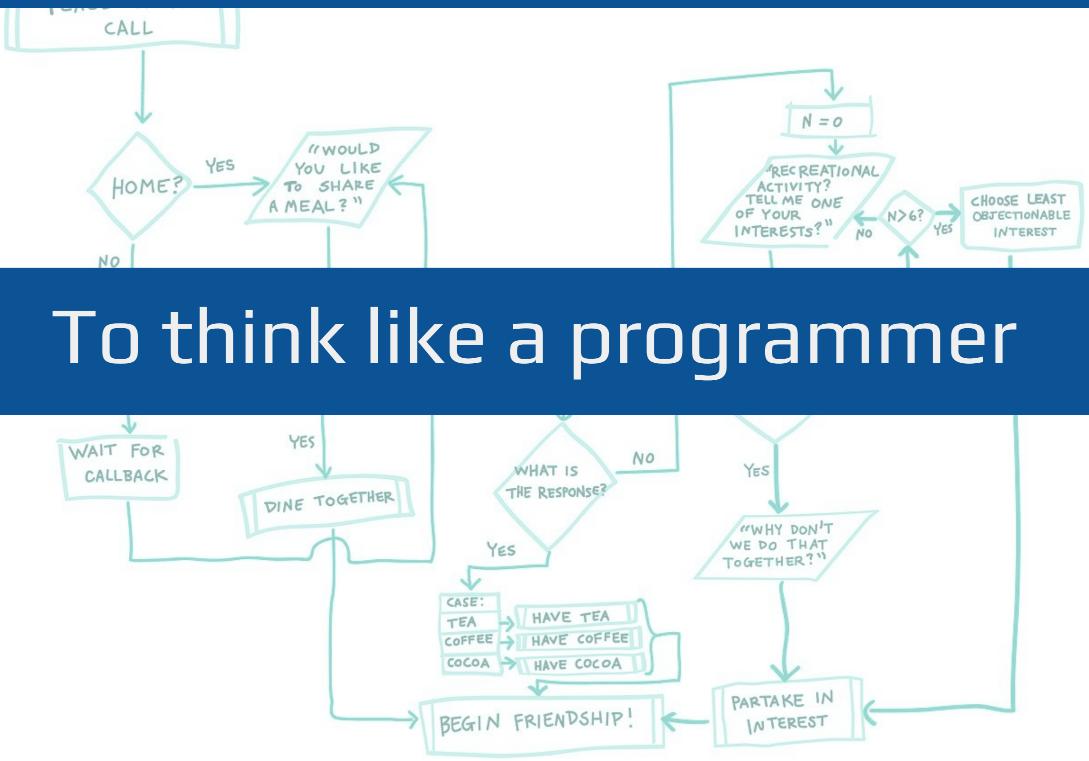


```python
# this is a comment
print("EScAPe frOm mALvASia")

treasure_chest = ['dagger','gold','bread']
inventory = []

sections = [("""
     You wake up in a dark, misty forest.
     Your head aches. You remember every little bit of last night's battle.
     They won. That's all you care about. That's all it matters...
     You remember everything then. Even your friend Skian calling your name
     when the Mac'raees attacked fast and furiously. Can you remember your
     NAME?""", "middle_forest"), #0
     ("""
          You are in the gate of what seems to be a small village at the edge of the forest.
          There is one GUARD at the gate holding a dagger. He seems serious though not
          particularly dangerous. You have to pass the gate and enter the village.""", "gate"), #1
     ("""
       You entered the pirates' guild. Smell of cheap rum, blood, and moist.
       Lots of noise. There is the barman and two TABLES of pirates. The one
       company sits in silence looking at each other. The other table is noisy. The exit is to the south.
       """, "pirate's guild"), #2
       ]

sec = {}
for counter in enumerate(sections):
    # print(counter)
    index = counter[0]
    long_name = counter[1][1] # indexed[1][0] is the description
    short_name = ''
    for C in long_name:
        if C in ' /_': # spaces and slashes to dashes
            short_name = short_name + '-'
        elif not C in ".'": # don't use periods and apostrophes
            short_name += C.lower() # lowercase
    sec[short_name] = index

dirs = { 0: ("North", "N", "n"),
         1: ("South", "S", "s"),
         2: ("West", "W", "w"),
         3: ("East", "E", "e")}
        # 'northeast': 4, 'ne': 4,
        # 'southeast': 5, 'se': 5,
```



```python
# this is a comment
print("EScAPe frOm mALvASia")

treasure_chest = ['dagger','gold','bread']
inventory = []

sections = [("""
          You wake up in a dark, misty forest.
          Your head aches. You remember every little bit of last night's battle.
          They won. That's all you care about. That's all it matters...
          You remember everything then. Even your friend Skian calling your name
          when the Mac'raees attacked fast and furiously. Can you remember your
          NAME?""", "middle_forest"), #0
          ("""
               You are in the gate of what seems to be a small village at the edge of the forest.
               There is one GUARD at the gate holding a dagger. He seems serious though not
               particularly dangerous. You have to pass the gate and enter the village.""", "gate"), #1
          ("""
            You entered the pirates' guild. Smell of cheap rum, blood,
            Lots of noise. There is the barman and two TABLES of pirate
            company sits in silence looking at each other. The other ta
            """, "pirate's guild"), #2
            ]

sec = {}
for counter in enumerate(sections):
    # print(counter)
    index = counter[0]
    long_name = counter[1][1] # indexed[1][0] is the description
    short_name = ''
    for C in long_name:
        if C in ' /_': # spaces and slashes to dashes
            short_name = short_name + '-'
        elif not C in ".'": # don't use periods and apostrophes
            short_name += C.lower() # lowercase
    sec[short_name] = index

dirs = { 0: ("North", "N", "n"),
         1: ("South", "S", "s"),
         2: ("West", "W", "w"),
         3: ("East", "E", "e")}
        # 'northeast': 4, 'ne': 4,
        # 'southeast': 5, 'se': 5
```

Aim of this module



To think like a programmer

What is programming?

PROBLEM SOLVING

Al • go • rithm

CALL
HOME?
YES
"WOULD YOU LIKE TO SHARE A MEAL?"
N = 0
"RECREATIONAL ACTIVITY? TELL ME ONE OF YOUR INTERESTS?"
N>6?
NO
YES
CHOOSE LEAST OBJECTIONABLE INTEREST
YES
WE DO THAT TOGETHER?"
CASE:
TEA
COFFEE
COCOA
HAVE TEA
HAVE COFFEE
HAVE COCOA
BEGIN FRIENDSHIP!
PARTAKE IN INTEREST

Discrete set of steps for a computer program to accomplish a task

What will we learn?

HOW TO CODE USING PYTHON

APPLY PRINCIPLES OF PROGRAMMING

WORK WITH IMPERATIVE &
OBJECT-ORIENTED PARADIGMS

BUILD GAMES BY programming

Python

- Uses an elegant syntax, making the programs you write easier to read.
- Is an easy-to-use language that makes it simple to get your program working.
- Comes with a large standard library that supports many common programming tasks
- Python's interactive mode makes it easy to test short snippets of code.

print ("Hello Python. Let's make a game")

Programming Paradigms

TERM 1: Imperative Paradigm

‘First do this, then do that’

TERM 2: Object-oriented Paradigm

Modularity with objects

Assessments

Imperative Paradigm (45%) Due 28/01/2019

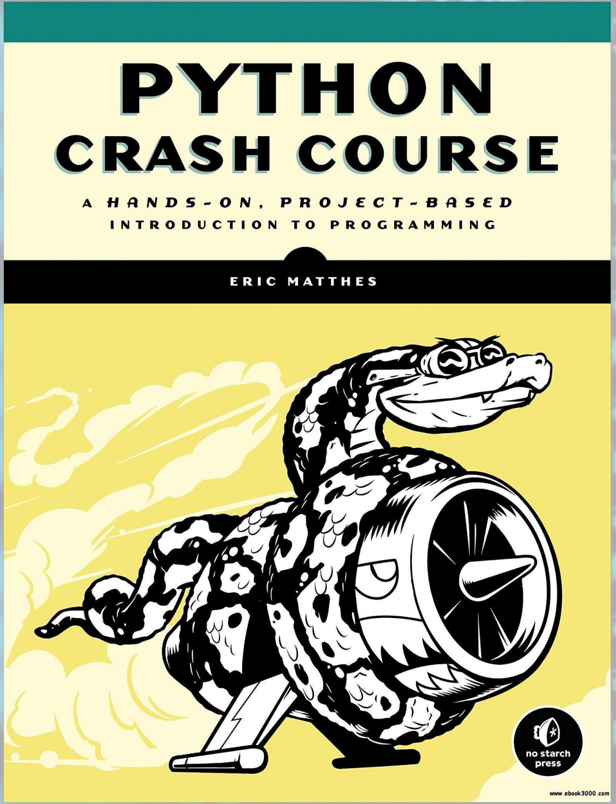
Text based adventure game

Object-oriented Paradigm (45%) Due 06/05/2019

Computer game

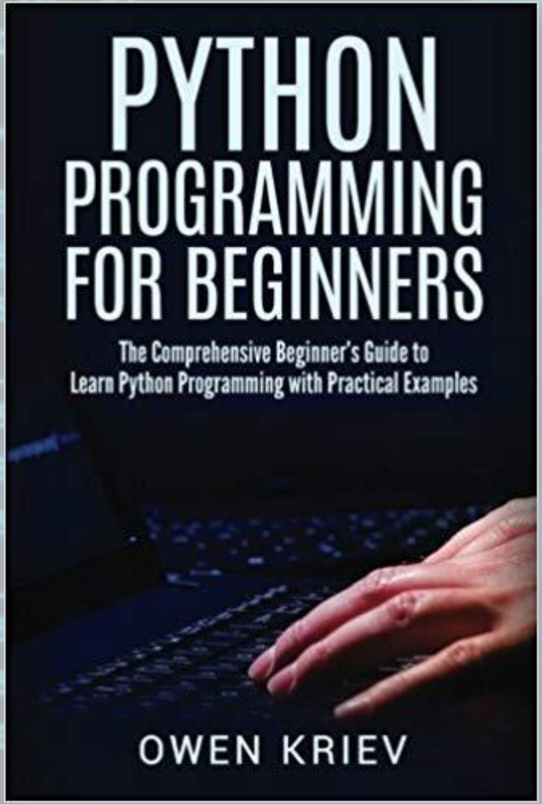
Participation (10%) Best 8 performances

Books!


PYTHON CRASH COURSE

A HANDS-ON, PROJECT-BASED INTRODUCTION TO PROGRAMMING

ERIC MATTHES


PYTHON PROGRAMMING FOR BEGINNERS

The Comprehensive Beginner's Guide to Learn Python Programming with Practical Examples

OWEN KRIEV

Tools

GitHub
Commits

print ("Hello Python. Let's make

Python Syntax

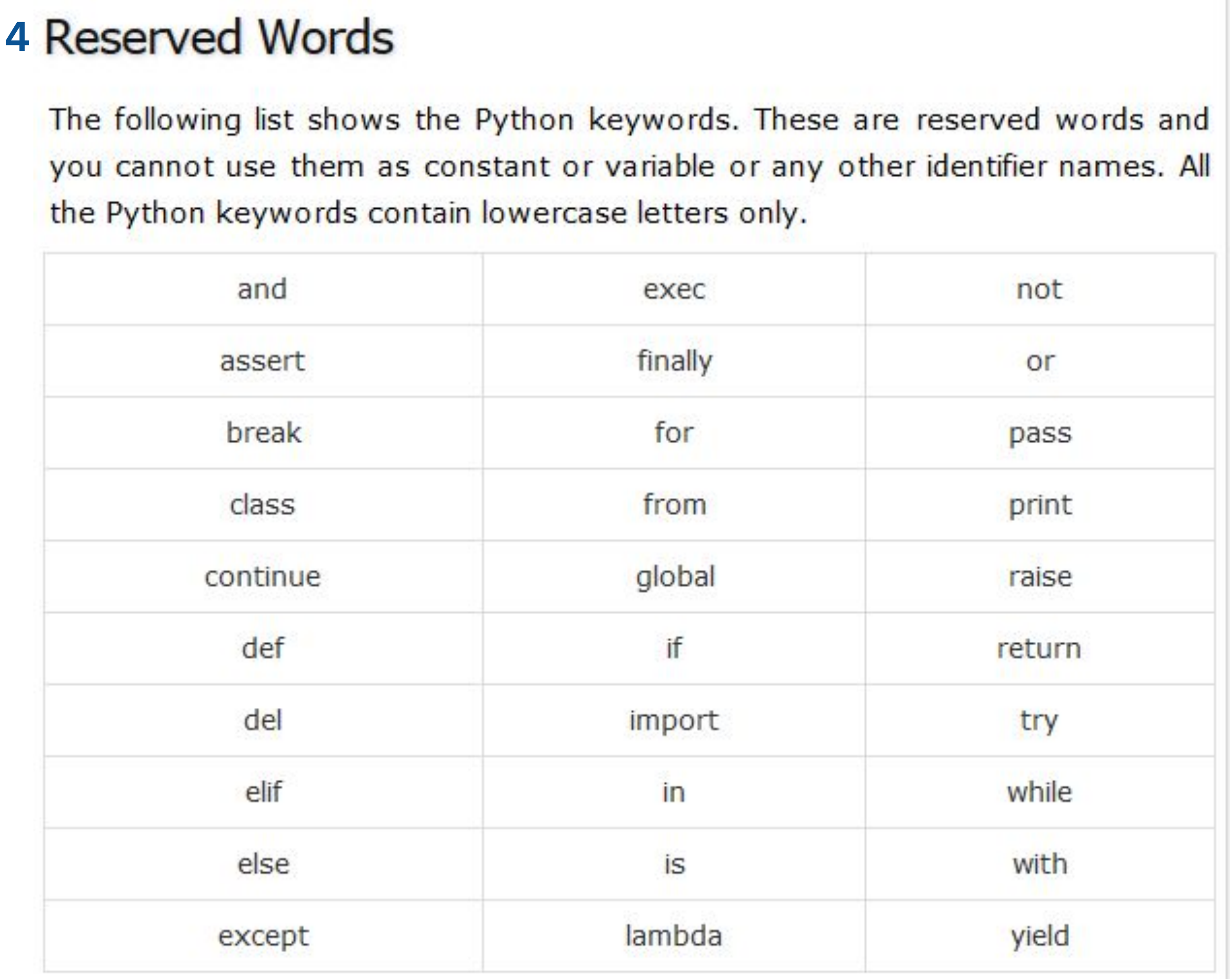
1 **Interactive mode** uses the Python Shell

2 **Script mode** uses a text editor.

3 **Indentation is important**

! WE WILL BE WORKING ON SCRIPT MODE WITH 'ATOM' TEXT EDITOR

Python Syntax



4 Reserved Words

The following list shows the Python keywords. These are reserved words and you cannot use them as constant or variable or any other identifier names. All the Python keywords contain lowercase letters only.

and	exec	not
assert	finally	or
break	for	pass
class	from	print
continue	global	raise
def	if	return
del	import	try
elif	in	while
else	is	with
except	lambda	yield

Python Syntax

5 Comments in Python

A hash sign (#) that is not inside a string literal begins a comment. All characters after the # and up to the end of the physical line are part of the comment and the Python interpreter ignores them.

```
# First comment
print "Hello, Python!" # second comment
```

Variables

Variables can hold values of different data types

```
i = 5
name = “My name”
i = i + 1
```

Data types

Literal Constants: 2, ‘A string!’, 45.6778

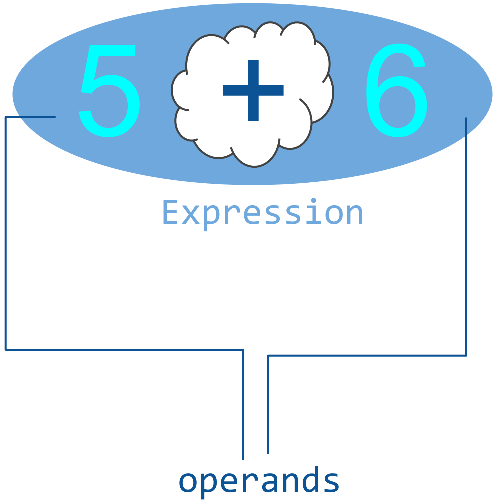
Numbers: integers and floats (**int, float**)

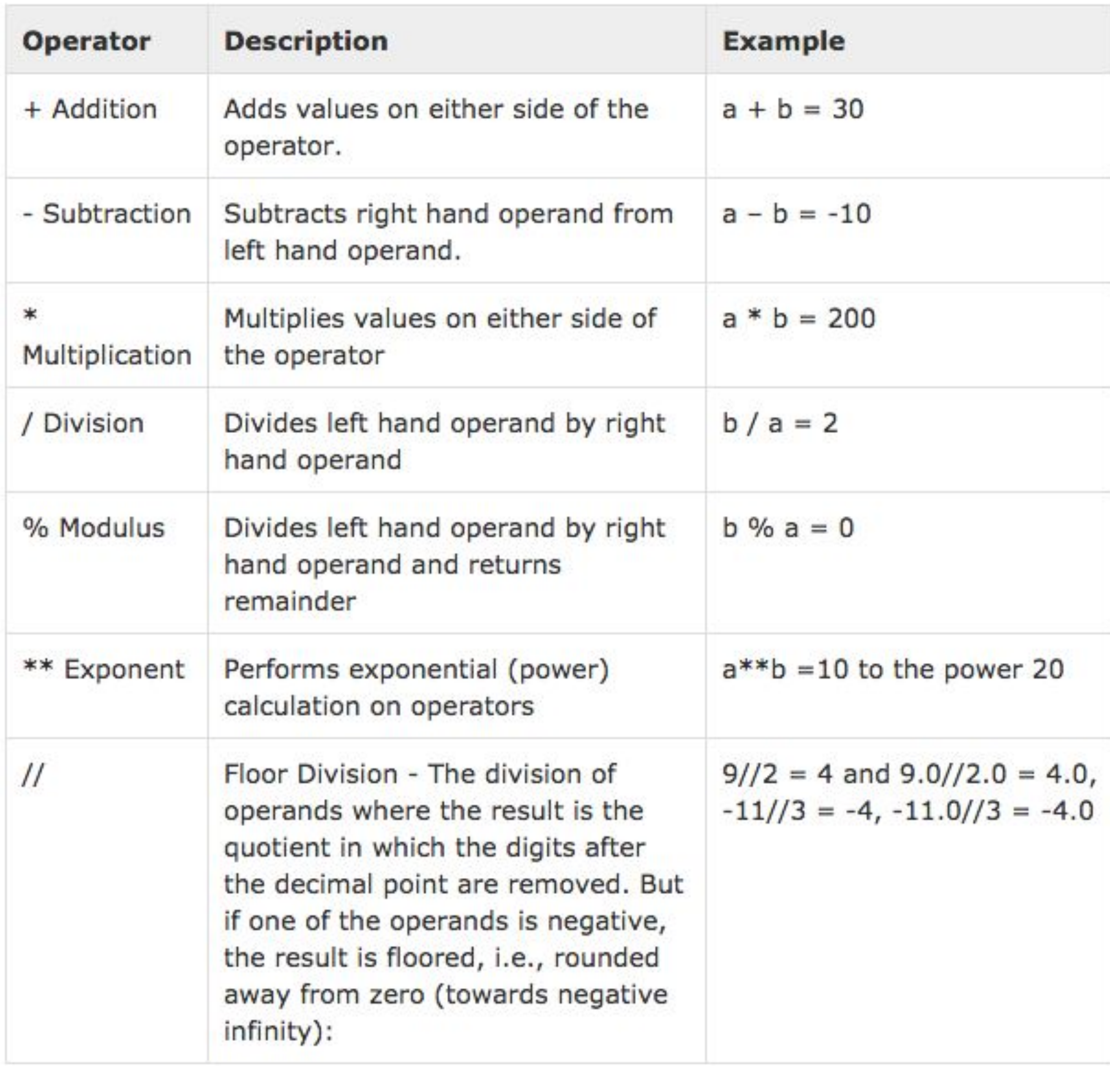
Strings (a sequence of characters): “This is a string in double quotations”, ‘This is a string in single quotations’

! STRINGS ARE IMMUTABLE. ONCE YOU DECLARE THEM YOU CANNOT CHANGE THEM!

[Python also has Lists, Tuples and Dictionaries but we will talk about them in next classes]

Arithmetic Operators





Operator	Description	Example
+ Addition	Adds values on either side of the operator.	a + b = 30
- Subtraction	Subtracts right hand operand from left hand operand.	a – b = -10
* Multiplication	Multiplies values on either side of the operator	a * b = 200
/ Division	Divides left hand operand by right hand operand	b / a = 2
% Modulus	Divides left hand operand by right hand operand and returns remainder	b % a = 0
** Exponent	Performs exponential (power) calculation on operators	a**b =10 to the power 20
//	Floor Division - The division of operands where the result is the quotient in which the digits after the decimal point are removed. But if one of the operands is negative, the result is floored, i.e., rounded away from zero (towards negative infinity):	9//2 = 4 and 9.0//2.0 = 4.0, -11//3 = -4, -11.0//3 = -4.0

Operators' order of precedence

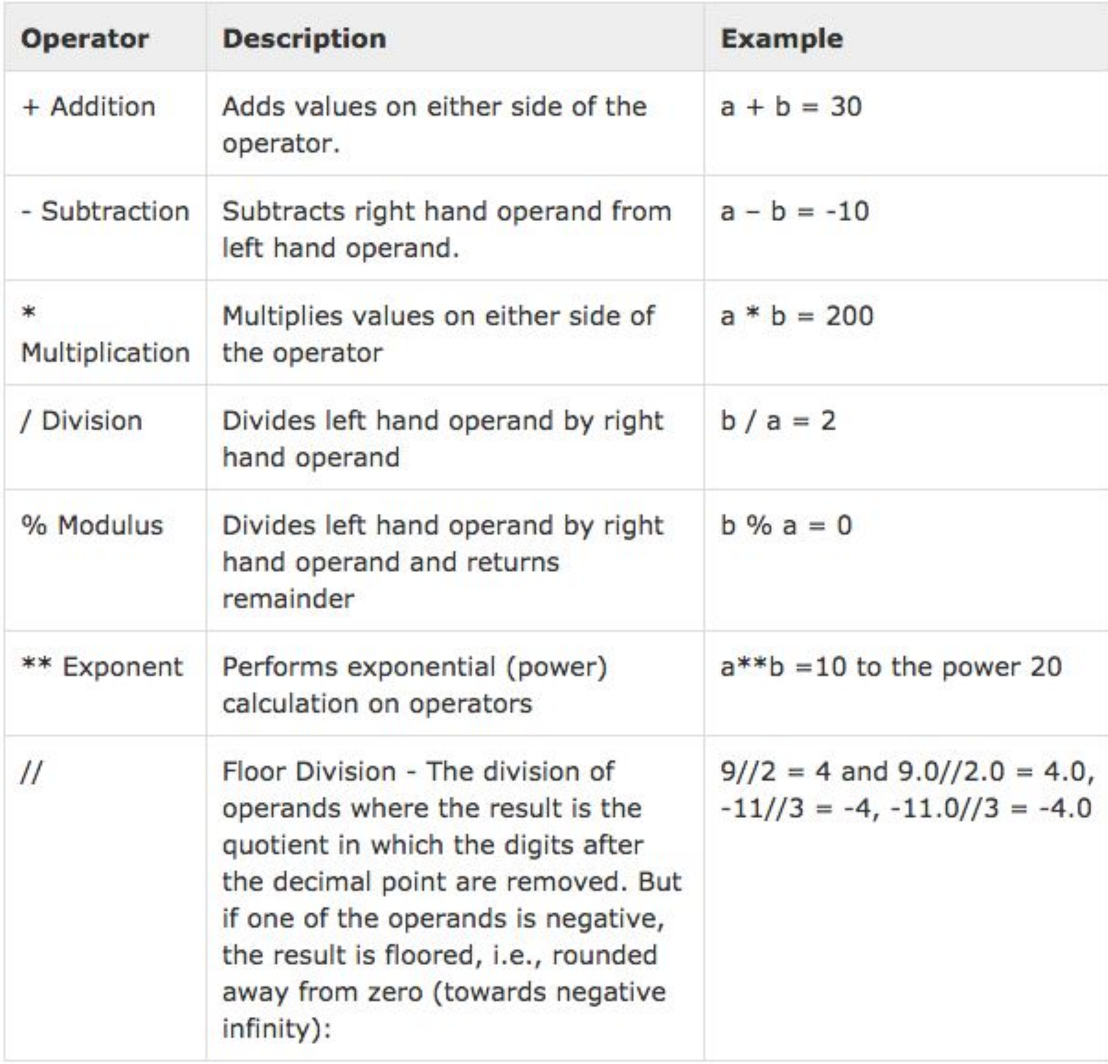
**

* , /, //, %

+ , -

Or use **parentheses** to change the order of evaluation

! Operators are usually associated from left to right. Operators with the same precedence are evaluated in a left to right manner. For example,
2 + 3 + 4 is evaluated as
(2 + 3) + 4.

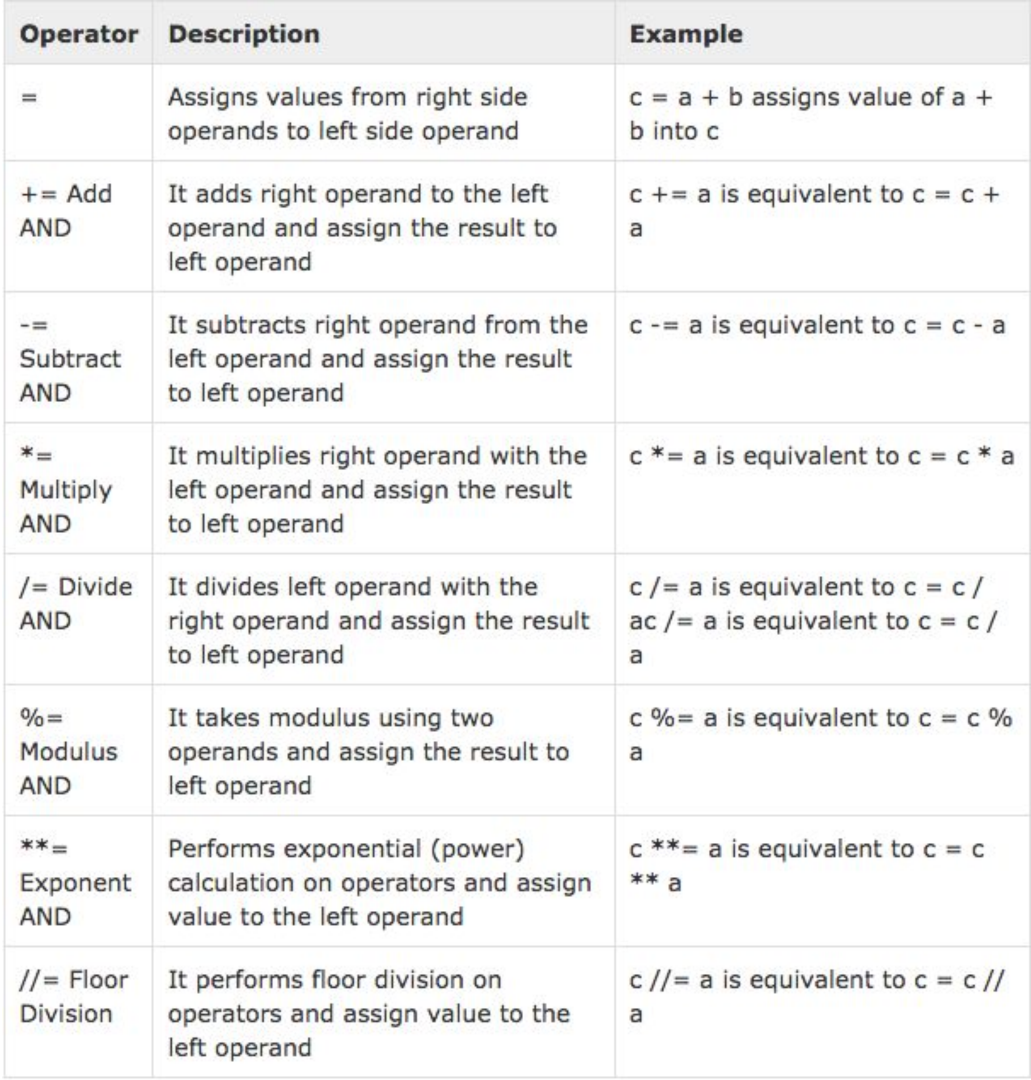


Operator	Description	Example
+ Addition	Adds values on either side of the operator.	a + b = 30
- Subtraction	Subtracts right hand operand from left hand operand.	a – b = -10
* Multiplication	Multiplies values on either side of the operator	a * b = 200
/ Division	Divides left hand operand by right hand operand	b / a = 2
% Modulus	Divides left hand operand by right hand operand and returns remainder	b % a = 0
** Exponent	Performs exponential (power) calculation on operators	a**b =10 to the power 20
//	Floor Division - The division of operands where the result is the quotient in which the digits after the decimal point are removed. But if one of the operands is negative, the result is floored, i.e., rounded away from zero (towards negative infinity):	9//2 = 4 and 9.0//2.0 = 4.0, -11//3 = -4, -11.0//3 = -4.0

Assignment Operators

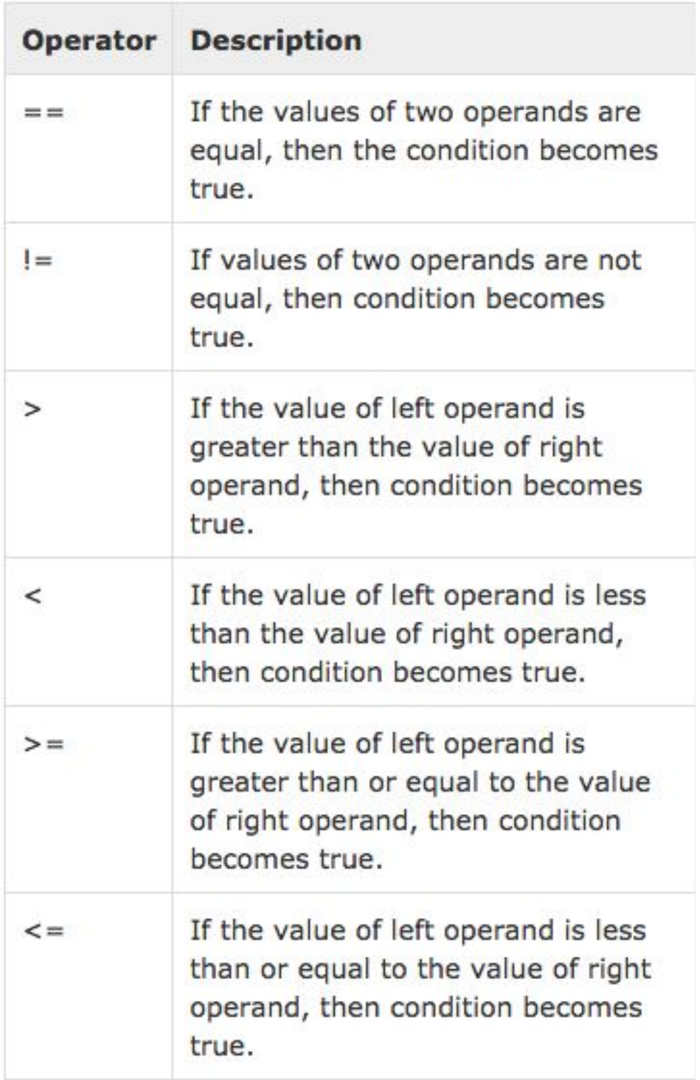
Problem solving I: You are in a game store in the US where prices are shown without tax. You are ready to buy a board game which costs $14 and there is a 12% tax on top. Write code to calculate how much you will buy it eventually.

Problem solving II: Your favorite game costs £20 but this week is on sale with 40% discount. Write code to calculate how much you will buy it eventually.



Operator	Description	Example
=	Assigns values from right side operands to left side operand	c = a + b assigns value of a + b into c
+= Add AND	It adds right operand to the left operand and assign the result to left operand	c += a is equivalent to c = c + a
-= Subtract AND	It subtracts right operand from the left operand and assign the result to left operand	c -= a is equivalent to c = c - a
*= Multiply AND	It multiplies right operand with the left operand and assign the result to left operand	c *= a is equivalent to c = c * a
/= Divide AND	It divides left operand with the right operand and assign the result to left operand	c /= a is equivalent to c = c / ac /= a is equivalent to c = c / a
%= Modulus AND	It takes modulus using two operands and assign the result to left operand	c %= a is equivalent to c = c % a
**= Exponent AND	Performs exponential (power) calculation on operators and assign value to the left operand	c **= a is equivalent to c = c ** a
//= Floor Division	It performs floor division on operators and assign value to the left operand	c //= a is equivalent to c = c // a

Comparison Operators



Operator	Description
==	If the values of two operands are equal, then the condition becomes true.
!=	If values of two operands are not equal, then condition becomes true.
>	If the value of left operand is greater than the value of right operand, then condition becomes true.
<	If the value of left operand is less than the value of right operand, then condition becomes true.
>=	If the value of left operand is greater than or equal to the value of right operand, then condition becomes true.
<=	If the value of left operand is less than or equal to the value of right operand, then condition becomes true.

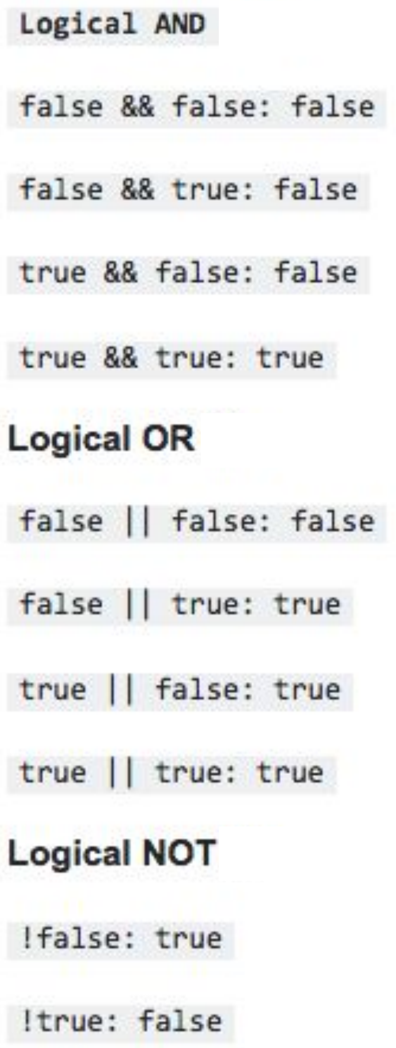
Logical Operators

In order of precedence:

NOT

AND

OR



`Logical AND`

`false && false: false`

`false && true: false`

`true && false: false`

`true && true: true`

Logical OR

`false || false: false`

`false || true: true`

`true || false: true`

`true || true: true`

Logical NOT

`!false: true`

`!true: false`

Strings

! Or use double quotation "There's a snake..."

```
voice = 'There\'s a snake in my boot!'
print(voice)

voice = "There’s a snake in my boot!"
voice = '''There\'s a snake in my boot!'''
```

Python accepts single ('), double (") and triple (''' or """) quotes to denote string literals, as long as the same type of quote starts and ends the string.

```
print(voice + "!!!!!")
```

Strings

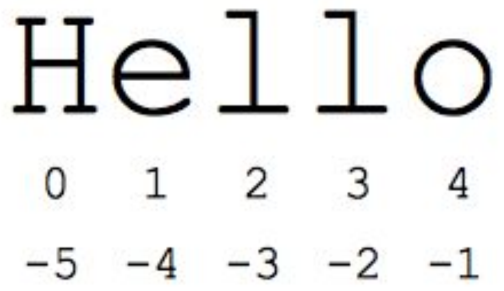
```
s = “hello”
print(s[1]) (answer : e)
print(s[1:3] (answer : el)
print(s[:2]) (answer : he)
print(s[-3:]) (answer : llo)
print(s[-1]) (answer: o)
print(len(s)) (answer : 5)
```

Prints the characters up to the character before the one with the last given index

So here we need to leave it blank so that it reaches the end of the string. -1 would give us the character with Index -2.

[Click for more about slicing strings]

INDEXING!



H	e	l	l	o
0	1	2	3	4
-5	-4	-3	-2	-1

Strings - the % operator

```
name = ‘My name’
print(“Your name is %s” % (name))


age = 12
height = 1.68
print(“Your name is %s, your age is %d and
you are %f cm tall” % (name, age, height))
```

For reducing the float to certain decimals, e.g for 2 decimals, write **%.2f**

Strings

s = “a string”

- **s.lower(), s.upper()** -- returns the lowercase or uppercase version of the string
- **s.strip()** -- returns a string with whitespace removed from the start and end
- **s.isalpha()/s.isdigit()/s.isspace()...** -- tests if all the string chars are in the various character classes
- **s.startswith('other'), s.endswith('other')** -- tests if the string starts or ends with the given other string
- **s.find('other')** -- searches for the given other string (not a regular expression) within s, and returns the first index where it begins or -1 if not found
- **s.replace('old', 'new')** -- returns a string where all occurrences of 'old' have been replaced by 'new'

User input

`input("text")` *for python3*

E.g. `name = input("What is your name? ")`

ATTENTION! THE RESULT OF INPUT() IS STRING!
YOU NEED TO DEFINE THE DATA TYPE OF THE INPUT.
E.G. FOR INT:

```
number = int( input("Give me a number :  ") )
```

`str``(number) will give you the number in string`

```
    E.g. str(2) is "2"
```

Flow control -Conditionals

If a then b else c

If a then b else if c then d … else e

* The *else* clause is optional.

INTENT!

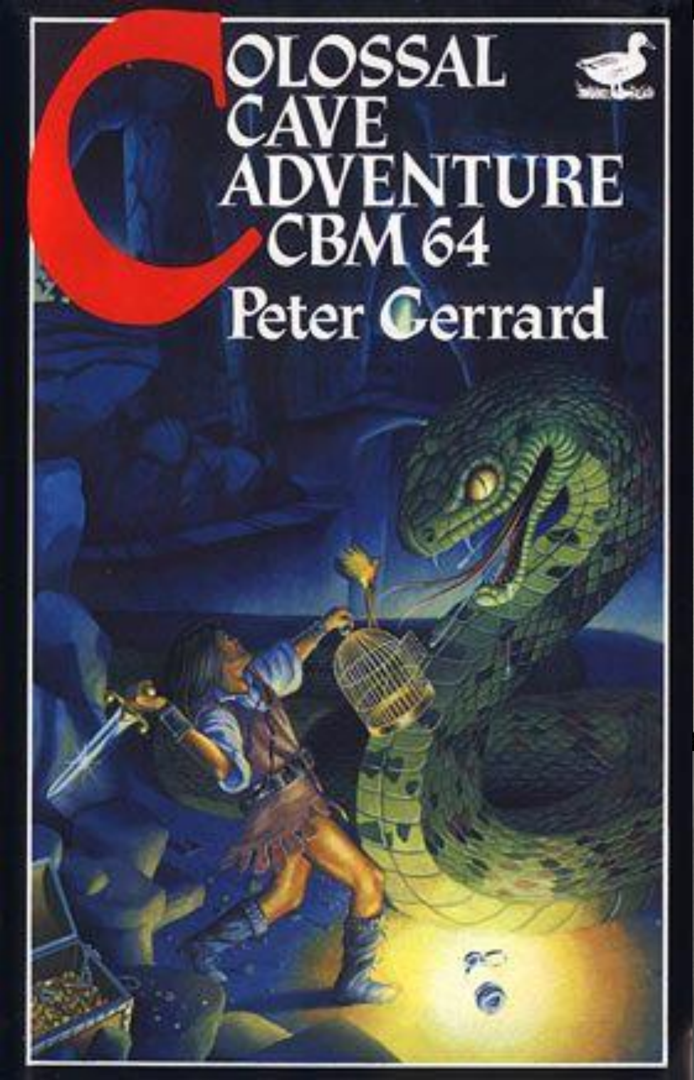
```
If a:
    Do something (b)
else:
    Do something else (c)
```

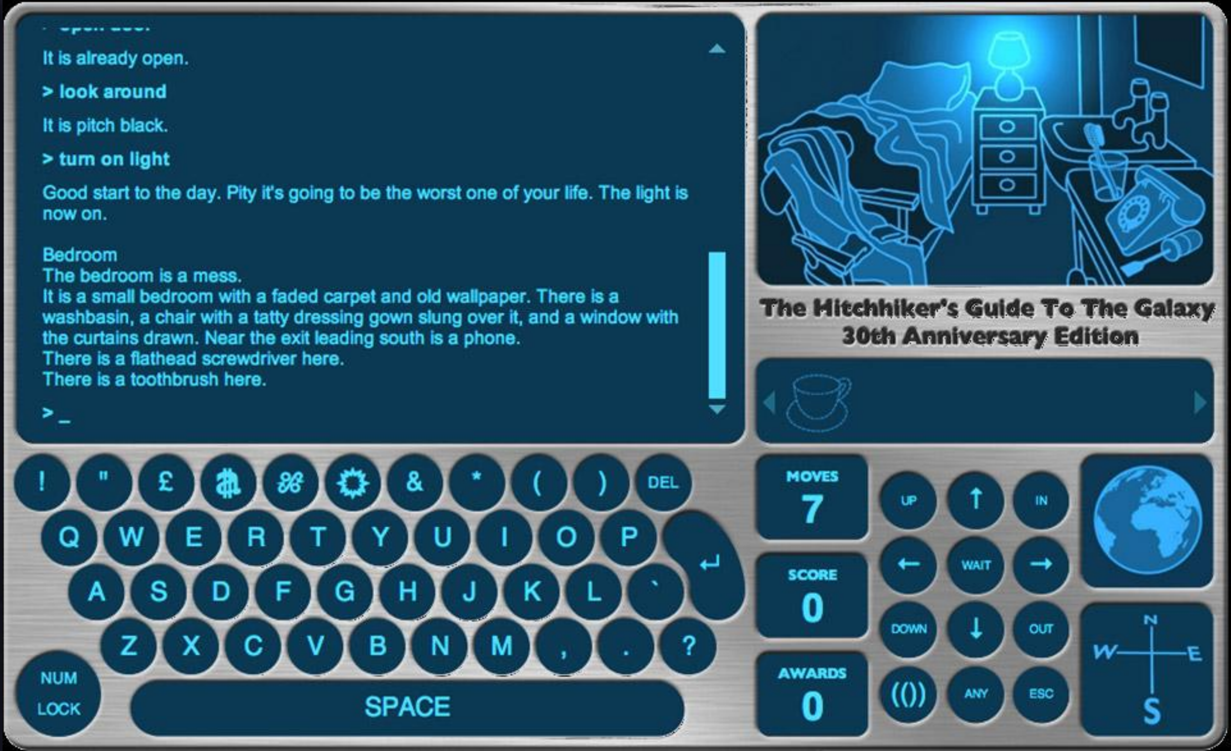
```
If a:
    Do something (b)
elif c:
    Do something else (d)
elif …:
    …
else:
    Do e
```

Conditionals

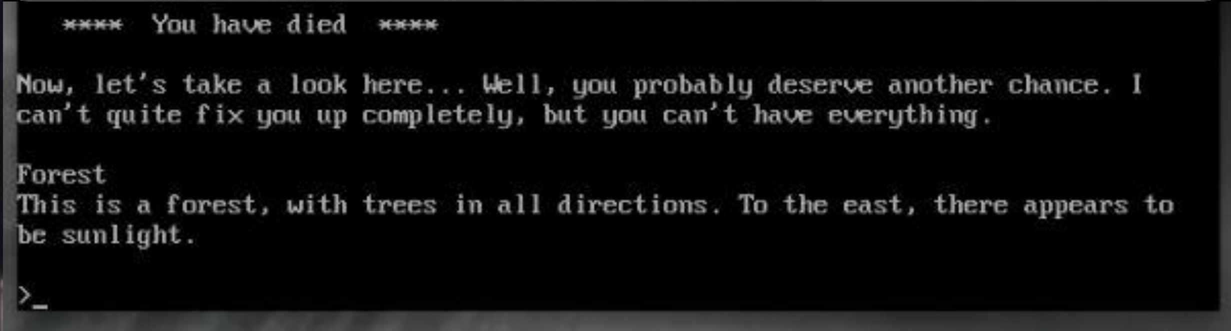
Problem solving III: Create a program that takes a number from user input and prints out :
“That’s a small number” if the number is equal or smaller than 100
“That’s not a big number yet” if the number is equal or smaller than 1000
“That’s an adequately huge number!” for any other number

Problem solving IV: Create a program that takes a number from user input and prints out :
“This is an odd number” if the number is odd
“This is an even number” if the number is even

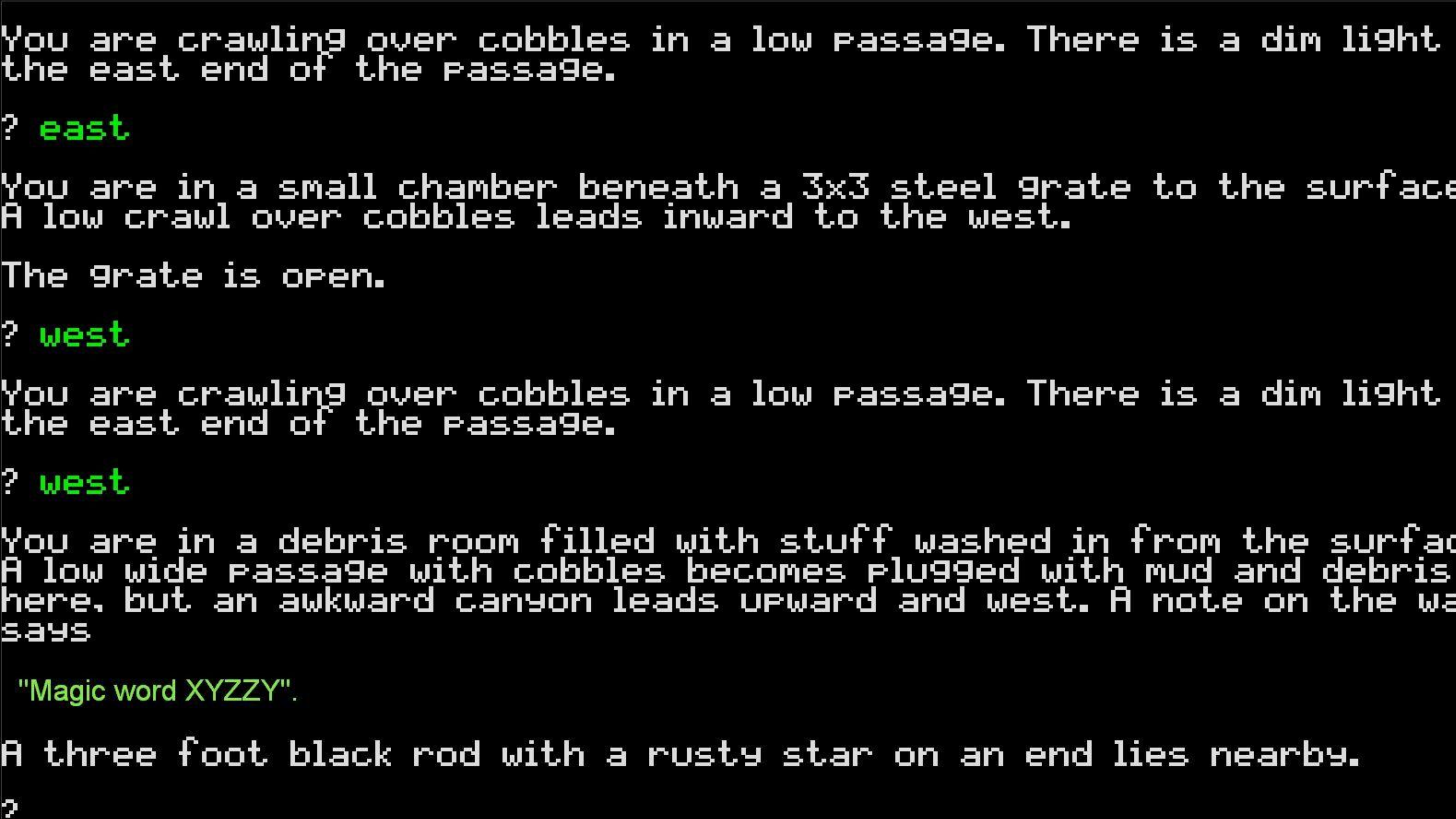




```
It is already open.

> look around

It is pitch black.

> turn on light

Good start to the day. Pity it's going to be the worst one of your life. The light is
now on.

Bedroom
The bedroom is a mess.
It is a small bedroom with a faded carpet and old wallpaper. There is a
washbasin, a chair with a tatty dressing gown slung over it, and a window with
the curtains drawn. Near the exit leading south is a phone.
There is a flathead screwdriver here.
There is a toothbrush here.

> _
```



```
  ****  You have died  ****

Now, let's take a look here... Well, you probably deserve another chance. I
can't quite fix you up completely, but you can't have everything.

Forest
This is a forest, with trees in all directions. To the east, there appears to
be sunlight.


>_
```



```
You are crawling over cobbles in a low passage. There is a dim light
the east end of the passage.

? east

You are in a small chamber beneath a 3x3 steel grate to the surface
A low crawl over cobbles leads inward to the west.

The grate is open.

? west

You are crawling over cobbles in a low passage. There is a dim light
the east end of the passage.

? west

You are in a debris room filled with stuff washed in from the surfa
A low wide passage with cobbles becomes plugged with mud and debris
here, but an awkward canyon leads upward and west. A note on the wa
says

 "Magic word XYZZY".

A three foot black rod with a rusty star on an end lies nearby.

?
```

1 MAP

2 NAVIGATION

3 OBJECTS YOU CAN ACT ON

4 OBJECTS YOU CANNOT ACT ON

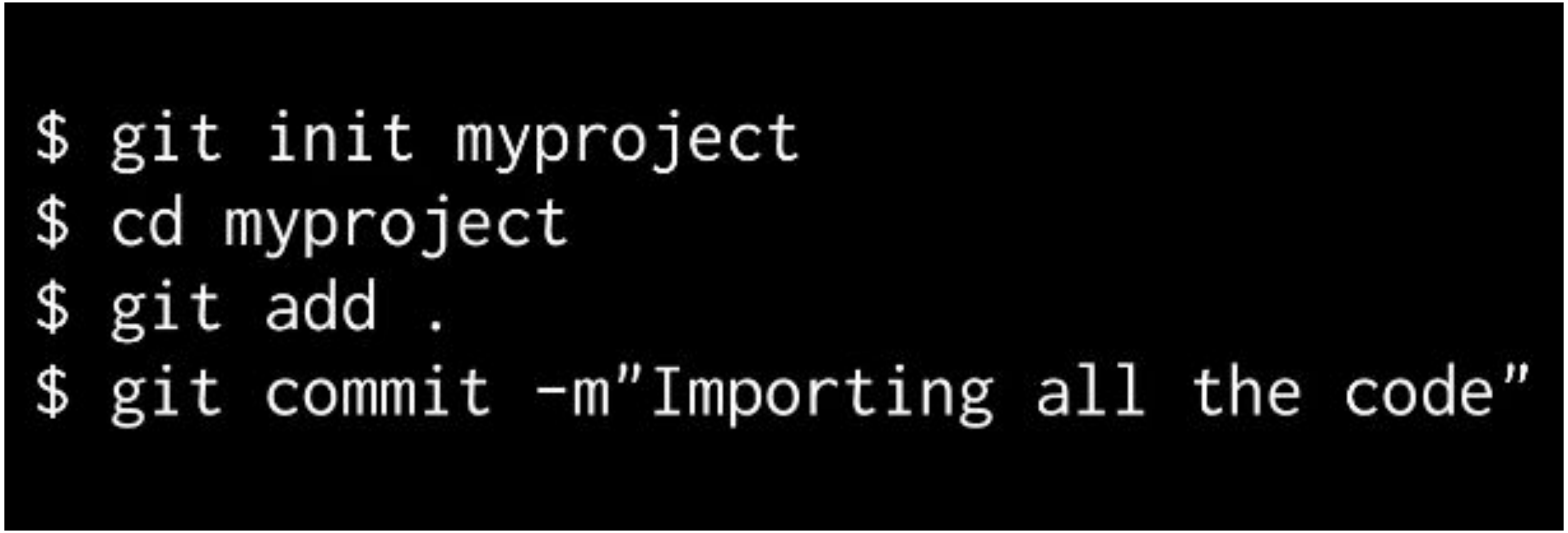
5 INVENTORY

6 VERB LIST

7 SAVE/LOAD GAME

Git is a source code version control tool. Version control is the …

Management of changes in your source code (across different collaborators)



```
$ git init myproject
$ cd myproject
$ git add .
$ git commit -m"Importing all the code"
```

GIT and Github

GitHub is a **<u>website</u>** where you can publish your Git repositories for public download and possible collaboration. **GitHub provides cloud services using Git.**

- You can work between your local repository and your github repository and work offline most of the time
- Each local copy contains the full repository's history

GIT - Configuration

Step 1: We create our remote repository on Github. Go to:

https://education.github.com/pack

and register, choosing username.

Step 2: Create a new repository (your remote). Give it a name and make it private.

We are now going to connect the remote repository with the local we just created so that any changes you make can be done on the remote too and you can also view easily the history of your changes. Go back to your command line showing the local project folder (local repository).

GIT - Configuration

Step 3 First you will configure Git with you username and email. Every git commit you do will include this information and we use the Github ones as we will work with Github.

```
git config --global user.name "your_Github_username"
```

```
git config --global user.email "your_Github_email"
```

GIT - Configuration

Step 4 Now you need to **connect** your local repository to the one you created on Github so that you can 'push' the work we do locally to the remote repo.

```
git remote add origin https://github.com/<username>/<repo name>.git
```

where <username> is your **GitHub account username** and <repo name> is **your Github repository name that you created**. Now check your Github repo.

You will have to use the whole URL just once. You can from now on use the word ***'origin'*** to add changes to your remote repository.

Now you can 'upload' the code you have added and committed on to your Github repo.

```
git push -u origin master
```

If you make changes to your local files you update your Gitub repo in the same way:

```
git add .
```

for all files

```
git add  file.py
```

for file-by-file

If you have files of same format you can also do `git add *._format` e.g. `git add *.txt` for text files

Second step is to commit the changes with a message (so we know what commit this is):

```
git commit -m "This is my first commit of file.py"
```

And then we 'push' it to the remote:

```
git push origin master
```

GIT - Configuration

If you want to get code from your **Github repo**, the command to do this is **‘git pull’** .

```
git pull origin master
```

Origin is the local name of your remote repo and master is its main branch, its trunk rather.
To check the status of changes and commits type:

```
git status
```

This will show you in red any files you have changed and not added.

Summary

MODULE AIM: PROGRAMMING SKILLS

ASSESSMENTS: PARADIGMS

TEXT BASED ADVENTURE GAME

Summary

OPERATORS AND EXPRESSIONS

DATA TYPES : numbers, strings

CONDITIONALS : if clause

GET USER INPUT

Homework

Problem solving V: Given a string, if its length is at least 3, add 'ing' to its end. Unless it already ends in 'ing', in which case add 'ly' instead. If the string length is less than 3, leave it unchanged. Print the resulting string.

Exploration of text adventures: Write a Python program that places the user in a space (a room, a forest etc.) and takes user input for navigation. Program a few steps from the first place to the next. At this stage the program does not have to be dynamic. Play with user input and strings.

Push your code on Github