**A PROJECT REPORT**

**ON**

**Session: 2020-21**

Pylock game

**FOR PRACTICAL FULLFILLMENT OF THREE YEAR’S DIPLOMA COURSE IN COMPUTER SCIENCE**

**Ambition Institute Of Technology, Varanasi**

**Uttar Pradesh**

**Under The Guidance Of Submitted By**

**Shreya Singh**

**Nishant Jaiswal**

**Aditya Mishra**

**Sakshee Upadhayay**

**Ashwit Upadhayay**

****

**CERTIFICATE**

This is to Certified that this project Report entitled **“PYLOCK GAME”** has been prepared by **SHREYA SINGH, NISHANT JAISWAL, ADITYA MISHRA, ASHWIT UPADHAYAY, SAKSHEE UPADHAYAY** of the Diploma Final Year during the Session 2020-2021 as a small project, a partial fulfilment for the award of the degree of diploma holder of the Board of Technical Education Lucknow Uttar Pradesh.

The project Report is up to the standard and I recommended to forward this project report to the controller of Examinations, Board of Technical Education, Lucknow Uttar Pradesh for getting it evaluated as per the ordinances governing the diploma course.

**Date: \_\_/\_\_/\_\_** **Project Manager:**

**Mr. Vivek Mishra Sir**

**Mr. S.K. Upadhayay Sir**

**ACKNOWLEDGEMENT**

**“No Learning is possible without proper Guidance”**

At every outset I express my gratitude to almighty lord for showering his grace and blessings upon us to complete project on **“PYLOCK GAME”**.

Although our name appears on the cover of this book, many people had contributed in some form or the other form to this project Development. We could not done this project without the assistance or support of each of the following we thank you all.

I wish to place on my record my deep sense of gratitude to my project guide, **Mr.Agha Alfi Mirza**, Information Technology Dept., for his constant motivation and valuable help through the project work. Express my gratitude to **Mr.Vivek Mishra**, Director of **Ambition Institute Of Technology** for his valuable suggestions and advices throughout the Computer course. I am highly thankful **Mr.Omprakash Gupta**, **Mr.Upendra Singh, Mr.Satish Gupta, Mr.Amit Singh, Mr.Alok Kumar Gupta, Mr.Shiv Govind** for his invaluable support and guidance.

Finally, I would like to thank all my friends for their cooperation to complete this project.

***Shreya Singh***

***Nishant Jaiswal***

***Aditya Mishra***

***Ashwit Upadhayay***

***Sakshee Upadhayay***

😊🎐🎐

**INDEX**

1. **CERTIFICATE**
2. **ACKNOWLEDEGMENT**
3. **INTRODUCTION**
4. **ABOUT PYTHON**
   1. **FUNCTIONS & FEATURES**
   2. **INSTALLATION**
5. **DIFFERENCE -2.x & 3.x**
6. **DATA TYPES**
7. **MUTABLE & IMMUTABLE OBJECTS**
8. **OPERATORS:**
   1. **Arithmetic**
   2. **Escape Sequence**
   3. **String**
   4. **Formatting Symbols**
9. **STRING PROCESSING**
10. **PROJECT ANALYSIS:**

* **ANACONDA**
* **CONDA**
* **SPYDER**
* **HARDWARE & SOFTWARE**

1. **LIBRARIES USED:** 
   1. **Pygame**
   2. **Random**
2. **PROJECT DESIGNING:**
   1. **Architecture of Pylock game**
   2. **Installing Pygame**
   3. **Codes of project.**
   4. **Snapshots**
3. **FUTURE SCOPE**
4. **CONCLUSION**
5. **BIBILIOGRAPHY**

**INTRODUCTION🡪**

**Tetris : Block shifting game**

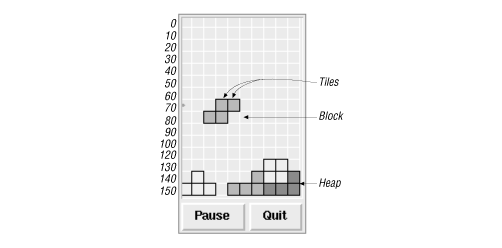
**PYTHON + BLOCK = “PYBLOCK GAME”** is aPython based block puzzle game which is popularly known as **Tetris game.** The game has a simple goal of destroying lines of block before it reaches the top. This puzzle game uses 4 square blocks joining edge to edge to form various combinations of shapes. There are 7 unique shapes. The position and rotation of shapes are controlled with the arrow keys of keyboard.

Benefits of playing puzzle game include problem solving skills, brain development, improved logical think, prioritizing things and much more.

***Tetris*** (Russian: Тетрис) is a [tile-matching video game](https://en.wikipedia.org/wiki/Tile-matching_video_game) created by Russian software engineer [Alexey Pajitnov](https://en.wikipedia.org/wiki/Alexey_Pajitnov) in 1984. It has been published by several companies, most prominently during a dispute over the appropriation of the game's rights in the late 1980s. After a significant period of publication by [Nintendo](https://en.wikipedia.org/wiki/Nintendo), the rights reverted to Pajitnov in 1996, who co-founded [The Tetris Company](https://en.wikipedia.org/wiki/The_Tetris_Company) with [Henk Rogers](https://en.wikipedia.org/wiki/Henk_Rogers) to manage *Tetris* licensing.

In *Tetris*, players must complete lines by moving differently shaped pieces (tetraamines), which descend onto the playing field. The completed lines disappear and grant the player points, and the player can proceed to fill the vacated spaces. The game ends when the playing field is filled. The longer the player can delay this inevitable outcome, the higher their score will be. In multiplayer games, the players must last longer than their opponents, and in certain versions, players can inflict penalties on opponents by completing a significant number of lines. Some adaptations have provided variations to the game's theme, such as three-dimensional displays or a system for reserving pieces.

**Tetris screen description**

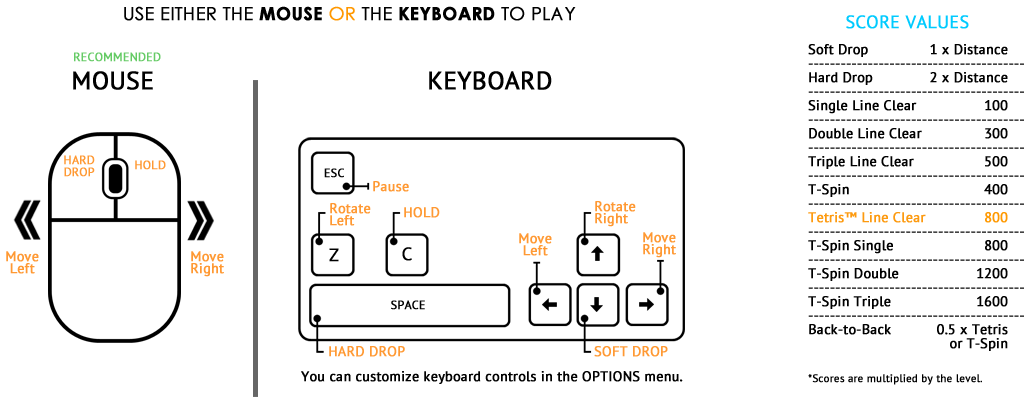


### [https://upload.wikimedia.org/wikipedia/commons/thumb/3/39/Tetrominoes_IJLO_STZ_Worlds.svg/220px-Tetrominoes_IJLO_STZ_Worlds.svg.png](https://en.wikipedia.org/wiki/File:Tetrominoes_IJLO_STZ_Worlds.svg)Game pieces

**How to play?**

The Tetris game requires players to strategically rotate, move, and drop a procession of Tetriminos that fall into the rectangular Matrix at increasing speeds. Players attempt to clear as many lines as possible by completing horizontal rows of blocks without empty space, but if the Tetriminos surpass the Skyline the game is over! It might sound simple, but strategy and speed can go a long way!

Use the Keyboard to Play:



**Python**

* It was initially designed by **Guido van Rossum in 1991.**
* It is most widely used general-purpose high-level programming language

(**Multi-programming Pradigm**).

* **Developed** by **Python** Software Foundation.
* It was mainly developed for **emphasis on code readability**, and its **syntax** allows programmers **to express concepts** in fewer lines of code.
* The **standard distribution** of Python (available on **python.org**) is developed in **C Language** and is often called **CPython**. This recommended standard is tweaked to **optimize** its **performance** for specific applications. These **implementations combine** ease of development of Python and **rich libraries** of other **platforms** such as **.net** or **Java**.

**Some such alternative implementations are** :

* **IronPython** (Python running on .NET framework. Developed in C#)
* **Jython** (Developed in Java. Python running on the Java Virtual Machine. Capable of using rich Java library in Python program)
* **PyPy** (A fast python implementation with a JIT compiler)
* **Stackless Python** (Branch of CPython supporting microthreads )
* **MicroPython** (Python running on micro controllers)

**Functions Of Python :**

* To create **web applications**.
* To create **workflows**.
* To connect to **database systems** (can **read** and **modify files**).
* To handle **big data** and perform **complex mathematics**.
* For **rapid prototyping.**
* For **production-ready software** development.

**Features :**

* Python works on **different platforms** (Windows, Mac, Linux, Raspberry Pi, etc.).
* (**Enhanced Readability**) Python has a **simple syntax** similar to the English language. Python has syntax that allows developers to write programs with fewer lines than some other programming languages.

**C/C++/JAVA**

if (X%2==0)

{ if (x%3==0) System.out.println("divisible by 2 and 3");

Else System.out.println("divisible by 2 not divisible by 3"); }

Else { if (x%3==0) System.out.println("divisible by 3 not divisible by 2");

Else System.out.println("not divisible by 3 nor by 2"); }

**Python**

if num%2 == 0:

if num%3 == 0: print ("Divisible by 3 and 2") else:

print ("Divisible by 2 not divisible by 3")

else: if num%3 == 0:

print ("Divisible by 3 not divisible by 2")

else: print ("Not divisible by 3 nor by 2")

* Python **runs on** an **interpreter system**, meaning that code can be executed as soon as it is written. This means that **prototyping can be very quick**.
* Python can be **treated** in a procedural way, **an object-orientated** way or a **functional way**.
* **Dynamic typing**: In Python, a variable to be used in a program need not have prior declaration of its name and type. You will learn details about variables in the next module.

**Java is statically typed**

String var; // Variable is explicitly declared.

Var = ”Hello”;

Var = 1234; // Not allowed because the variable has been declared as a String.

**Python is dynamically typed**

var= ”Hello” # Variable is NOT explicitly declared.

var=1234 # Allowed although variable was initially String type

* **Interpreted language**:

Python is an interpreter-based language. It executes one instruction at a time. So if there is an error on line 7 of the code, it will execute instructions till line 6 and then stop. This is unlike a compiler which will not execute any of the instructions even if there is a single error. Thus, an interpreter is useful if you are new to programming because it allows you to see partial output of your code and identify the error location more easily

**Python :**

print ("welcome to")

print ("Python training program from")

print (XY Industries)

**Java :**

public class test

{

public static void main(String args[])

{

System.out.println("Welcome to"); System.out.println("Python training program");

System.out.println(From Stark );

}

}

* **Extensible language**:

Python extension modules implement new built-in object types and can call C libraries and system calls which Python doesn’t have direct access to.

* **Standard DB2 API** :

A standard data connectivity API facilitates using data sources such as Oracle, MySQL and SQLite as a backend to a Python program for storage, retrieval and processing of data.

* **GUI programming** :

Standard distribution of Python contains Tkinter GUI kit, which is an implementation of the popular GUI library called Tcl/Tk. An attractive GUI can be constructed using Tkinter. Many other GUI libraries like Qt, GTK, WxWidgets etc. are also ported to Python.

* **Embeddable :**

Python can be integrated with other popular programming technologies such as C, C++, Java, ActiveX and CORBA.

**Installation of Python :**

**Using the Package Manager**:

* 1. If you are using Ubuntu Linux, at the command prompt type:

**$ Sudo apt-get install python3**

* 1. If you are using Fedora, at the command prompt type:

**$ Sudo yum install python3**

* 1. The most recent version of Python 3 will be downloaded and installed. To verify the installation, type:

**$ Python**

The Python prompt (**>>>**) will appear.

**Difference between Python 2.x and Python 3.x:**

1. Parentheses are mandatory with the print function. Valid statement

**print ("Hello World")**

If the parentheses are missing, the interpreter displays the following error:

**SyntaxError: Missing parentheses in call to 'print'**

1. The raw\_input() function has been deprecated and input() function more or less behaves as raw\_input() thereby input is always read as string.
2. Evaluates the result of a division as decimals even if the input number are not specified with decimals.

**>>> 3/2**

**Output = 1.5**

1. Stores strings as Unicode by default.
2. Integer objects are long by default and do not require the trailing L.

**Example: 100**

1. It is not mandatory to use parenthesis with the in-built print function. Valid statements

**print "Hello World"**

**print ("Hello World")**

1. Two types of input functions are available to read data from the console. The raw\_input() function always treats the input as string, while the input() function evaluates the input and accordingly decides the data type of variable.
2. Numbers that you type without any digits after the decimal point are treated as integers during division.

**>>> 3/2**

**Output = 1**

1. Strings are stored as ASCII by default. To store strings as Unicode, they have to be marked with a ‘u’.

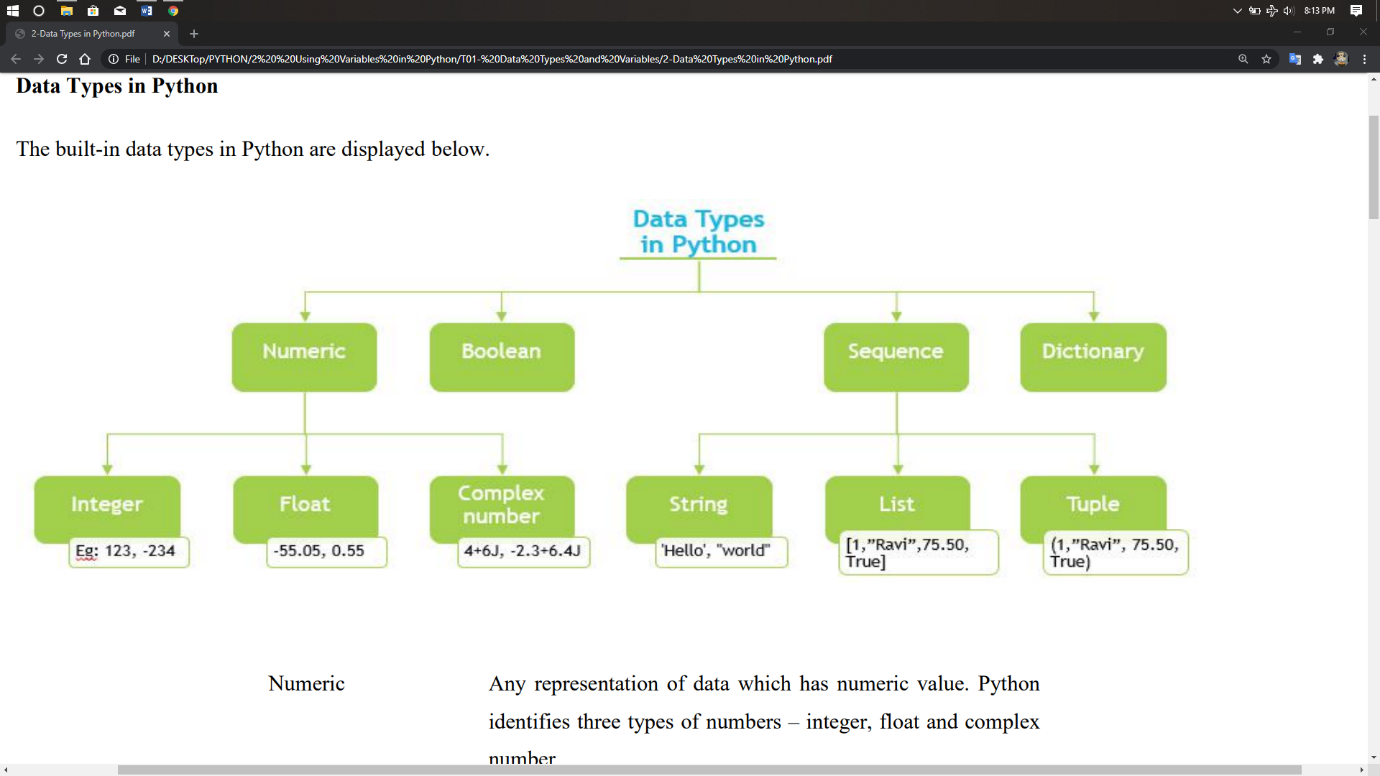
**Example: u’hello’**

1. Arithmetic operations over normal integers may overflow the memory allocated to them. To allocate more memory to an integer object, a trailing L is attached.

**Example: 100L**

**Data Types in Python :**

The built-in data types in Python are displayed below.



1. **Numeric:** Any representation of data which has numeric value. Python identifies three types of numbers – integer, float and complex number.
2. **Integer:** Positive and negative whole numbers.

**Examples:** 1234, -234, 0x46 (hexadecimal number), 0O123 (octal number))

**Note:** In C and related programming languages such as Python, a hexadecimal number is prefixed with 0x and an octal number is prefixed with 0O.

1. **Float:** Real numbers with a floating point representation in which the fractional component is denoted by a decimal or scientific notation.

**Examples:** -55.550, 0.005, 1.32E10 (scientific notation))

1. **Complex number:** A number with a real and imaginary component is represented as a + bj in Python where a and b are floats and j = √-1

**Examples:** 4+6j, -2.3+6.4j Note: The common mathematical representation of a complex number uses a +bi with i being the imaginary part. But in electronics j is used because i already represent current and the next letter after i is j.

1. **Boolean:** Any representation of data which has two values denoted by True and False.
2. **Sequence:** An ordered collection of similar or different data types. The built-in Sequence data types in Python are – String, List and Tuple.
   1. **String:** A collection of one or more characters put in single, double or triple quotes.

**Examples:** ‘Hello’, "Hello", "'Hello'", """Hello"""

* 1. **List:** An ordered collection of one or more data items, not necessarily of same type, put in square brackets.

**Examples:** [1,"Ravi", 75.50 , True]

* 1. **Tuple:** An ordered collection of one or more data items, not necessarily of same type put in parentheses. The contents of a tuple cannot be modified

– It is immutable - after the tuple is created.

**Examples**: (1,"Ravi", 75.50, True) Note: Refer to the Helper Text to learn more about mutability.

* 1. **Dictionary:** An unordered collection of data in key:value pair form. Collection of such pairs is enclosed in curly brackets.

**Example:** {1:"Superman", 2:"Wonder Woman", 3:"Thor", 4: "Hulk", 5:"Black Widow"}

**Mutable and Immutable Objects :**

When a program is run, data objects in the program are stored in the computer’s memory for processing. While some of these objects can be modified at that memory location, other data objects can’t be modified once they are stored in the memory. The property of whether or not data objects can be modified in the same memory location where they are stored is called **mutability**.

**>>>**We can check the mutability of an object by checking its memory location before and after it is modified. If the memory location remains the same when the data object is modified, it means it is mutable.

To check the memory location of where a data object is stored, we use the function**, id()**.

Consider the following example (you can try this yourself in IDLE):

**# Assigning values to [A].**

**# Using id() function.**

**# Location where A is stored.**

**# Replacing the item of list.**

**# Verifying**

**# Values changed in list.**

**# Using Id() function.**

**# Location where A is stored.**

**>>> a=[5, 10, 15]**

**>>> id(a)**

**1906292064**

**>>> a[1]=20**

**>>> print(a)**

**[5, 20, 15].**

**>>> id(a)**

**1906292064**

**Note:**

**Immutable:** Numeric, String and Tuple

**Mutable:** List, Dictionary

**Arithmetic Operators :**

**Floor division:**

**Operator for Floor Division, //**

Returns 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).

**Example 1**

>>> a=9

>>> b=5

>>> c=a//b

>>> c

1

**Exponent:**

**Operator for Exponent, \*\***

Calculates the value of the operand on the left raised to operand on the right of the operator.

**Example 1**

>>> a=4

>>> b=3

>>> c=a\*\*b

>>> c

64

**Modulus:**

**Operator for Modulus, %**

Returns the remainder of division of the operand on the left by the operand on the right of the operator.

**Example 1**

>>> a=21

>>> b=10

>>> c=a%b

>>> c

1

**Division:**

**Operator for Division, /**

Divides left hand operand by right hand operand.

**Example 1**

>>> a= -21

>>> b=10

>>> c=a/b

>>> c

2.1

**Multiplication:**

**Operator for Multiplication, \***

Multiplies values on either side of the operator.

**Example 1**

>>> a=21

>>> b=10

>>> c=a\*b

>>> c

210

**Subtraction:**

**Operator for Subtraction, -**

Subtracts the operand on the right from the operand on the left.

**Example 1**

>>> a=21

>>> b=10

>>> c=a-b

>>> c

11

**Addition:**

**Operator for Addition, +**

Adds the operands on either side of the operator.

**Example 1**

>>> a=21

>>> b=10

>>> c=a+b

>>> c

31

**Escape Sequences**:

You can use two or more specially designated characters within a string to format a

string or perform a command. These characters are called **Escape sequences**.

An escape sequence in Python starts with a **backslash (\)**.

**For example :** **\n** is an escape sequence in which the common meaning of the letter n is literally escaped and given an alternative meaning - a new line.

I designed this rhyme to explain in due time

All I know

Time is a valuable thing

Watch it fly by\ as the pendulum swings.

It doesn't even matter how hard you try

It is so "unreal"

Sound effect is heard🔔🎐🔔

>>> print('I designed this rhyme to explain in due time\n All I know')

>>> print('Time is a \t valuable thing')

>>> print('Watch it fly by \\ as the pendulum swings')

>>> print('It doesn\'t even matter how hard you try')

>>> print('It is so \"unreal\"')

>>> print('\a')

Breaks into a new line.

Adds a horizontal tab.

Prints a backslash.

Prints a single quote.

Prints a double quote.

Makes a sound like a bell.

**\n**

**\t**

**\\**

**\'**

**\"**

**\a**

**String Operators:**

**in Membership:**

**[in] Returns true if a character exists in the given string.**

**Example:**

>>> a='Harry watched Dumbledore striding up and down in front of him.' #String a

>>> 'v' in a #Checks if the character 'v' is present in the string, aFalse

**Range Slice:**

**[ : ] Fetches characters in the range specified by two index operands separated**

**by a colon.**

If the first operand is omitted, the range starts at zero index.

If the second operand is omitted, the range goes up to the end of the string.

Note: The slice starts at the first index. The slice ends one index before the second

index, that is at the value of the index - 1.

**Example:**

T h e B u r r o w

0 1 2 3 4 5 6 7 8 9

>>> a='The Burrow' #String a

>>> a[2:7] #Starting index = 2 = e, Ending index = 7-1 = r'e Bur'

>>> a[:6] #Starting index = 0 = T, Ending index = 6-1 = u'The Bu

**Slice:**

**[]Gives the character at given index.**

**Example:**

T h e B u r r o w

0 1 2 3 4 5 6 7 8 9

>>> a='The Burrow' #String a

>>> a[8] #Here the index number in square brackets [ ] slices out the character at that index, which is o in this example.'o'

**Repetition:**

**[\*] Concatenates multiple copies of the same string.**

**Example:**

>>> a='I must not tell lies…' # String a

>>> print('Again and again Harry wrote the words on the parchment in his own blood

– '+a\*5) # Here, the \* joins or concatenates the string a repeatedly for 5 times.

Again and again Harry wrote the words on the parchment in his own blood – I must not tell

lies…I must not tell lies…I must not tell lies…I must not tell lies…I must not tell

lies…

**Concatenation:**

**[+] Appends the second string to the first.**

**Example:**

>>> a='"Hello, ' #String a

>>> b='Nick",' #String b

>>> print(a+b+ ' said Harry to Nearly Headless Nick.') #The + here appends the

strings a and b to the string in quotes.

"Hello, Nick", said Harry to Nearly Headless Nick.

**not in Membership:**

**[not in] Returns true if a character does not exist in the given string.**

**Example:**

>>> a=''' “For HIM?” shouted Snape. “Expecto Patronum!" From the tip of his wand burst the silver doe: She landed on the office floor, bounded once across the office, and soared out of the window. Dumbledore watched her fly away, and as her silvery glow faded, he turned back to Snape, and his eyes were full of tears. “After all this time?” “Always,” said Snape.''' #Multi-line string, a

>>> 'v' not in a #Checks that the character 'v' is not present in the string, a False

>>> 'Red' not in a #Checks that the characters 'Red' is not present in the string, a True

>>> 'red' not in a #This is case sensitive. Since 'red' is present in 'soared', this returns False.

**Format specification symbols:**

**%c 🡪 #**character

**%s 🡪 #**string conversion via str() prior to formatting

**%i 🡪 #**signed decimal integer

**%d 🡪 #**signed decimal integer

**%u 🡪 #**unsigned decimal integer

**%o 🡪 #**octal integer

**%x/ %X 🡪 #**hexadecimal

**The format() Method:**

With Python 3.x , the format() method has been introduced for handling complex string formatting more efficiently.

This method of the built-in string class provides functionality for complex variable substitutions and value formatting. This new formatting technique is regarded as more elegant. The general syntax of **format()** method is:

**string.format(var1, var2,...)**

The string itself contains placeholders {} in which values of variables are

successively inserted.

>>> name="Malhar"

>>> age=23

>>>percentage=55.5

>>> "my name is {} and my age is {} years"**.format**(name, age)

'my name is Malhar and my age is 23 years'

**String Processing:**

**capitalize():** This method converts the first character of a string to uppercase letter.

>>> var='training'

>>> var.capitalize()

'Training'

**index():** This method is similar to find() but throws a ValueError if the substring is not found.

>>> var='python training'

>>> var.index('n')

5

>>> var.index('run')

Traceback (most recent call last):

File "<pyshell#19>", line 1, in <module>

var.index('run')

ValueError: substring not found

**upper():** This method returns a string with lowercase characters replaced by

corresponding uppercase characters.

>>> var='internshala'

>>> var.upper()

'INTERNSHALA'

**title():** This method results in a string with the first character of each word converted

to uppercase.

>>> var='python training'

>>> var.title()

'Python Training'

**find():** This method finds the first occurrence of a substring in another string. If not found, the method returns -1.

>>> var='python training'

>>> var.find('n')

5

>>> var.find('on')

4

**isdigit():** This method returns true if all characters in a string are digits( 0-9), if not

returns false.

>>> var='2000'

>>> var.isdigit()

True

>>> var='2,000'

>>> var.isdigit()

False

**count():** This method returns the number of occurrences of a substring in given string.

>>> var='python training'

>>> var.count('in')

2

**lower():** This method results in a string with uppercase characters replaced by corresponding lowercase characters.

>>> var='TRAINING'

>>> var.lower()

'training'

**isalpha():** This method returns true if all the characters in a string are alphabetic (a-z or A-Z), otherwise returns false.

>>> var='Training'

>>> var.isalpha()

True

>>> var='Train ing'

>>> var.isalpha()

False

**islower():** This method returns true if all characters in a string are lowercase characters else returns false.

>>> var='training'

>>> var.islower()

True

>>> var='Training'

>>> var.islower()

False

>>> var='trai ning'

>>> var.islower()

True

**isupper():** This method returns true if all characters in a string are uppercase

characters else returns false.

>>> var='TRAI\_NING'

>>> var.isupper()

True

>>> var='TRAining'

>>> var.isupper()

False

>>> var='TRAI+NING'

>>> var.isupper()

True

>>> var='1234'

>>> var.isupper()

False

**ANACONDA**

**Anaconda** is a an open source distribution of the Python and R programming languages and it is used in data science, machine learning, deep learning-related applications  aiming at simplifying package management and deployment.It is Usedfor [scientific computing](https://en.wikipedia.org/wiki/Scientific_computing) ([data science](https://en.wikipedia.org/wiki/Data_science), [machine learning](https://en.wikipedia.org/wiki/Machine_learning) applications, large-scale data processing, [predictive analytics](https://en.wikipedia.org/wiki/Predictive_analytics), etc.), that aims to simplify [package management](https://en.wikipedia.org/wiki/Package_management) and deployment. Package versions are managed by the [package management system](https://en.wikipedia.org/wiki/Package_manager) [*conda*](https://en.wikipedia.org/wiki/Conda_(package_manager)). The Anaconda distribution is used by over 13 million users and includes more than 1400 popular data-science packages suitable for Windows, Linux, and MacOS .

**Anaconda distribution** comes with more than 1,400 packages as well as the [Conda](https://en.wikipedia.org/wiki/Conda_(package_manager)) package and virtual environment manager, called **Anaconda Navigator.**

The open source packages can be individually installed from the Anaconda repository  with the conda install command or using the pip install command that is installed with Anaconda. Pip packages  provide many of the features of conda packages and in most cases they can work together.

Custom packages can be made using the conda build command, and can be shared with others by uploading them to Anaconda Cloud,[PyPI](https://en.wikipedia.org/wiki/Python_Package_Index) or other repositories.

The default installation of Anaconda2 includes Python 2.7 and Anaconda3 includes Python 3.7. However, you can create new environments that include any version of Python packaged with conda



**Anaconda Navigator**

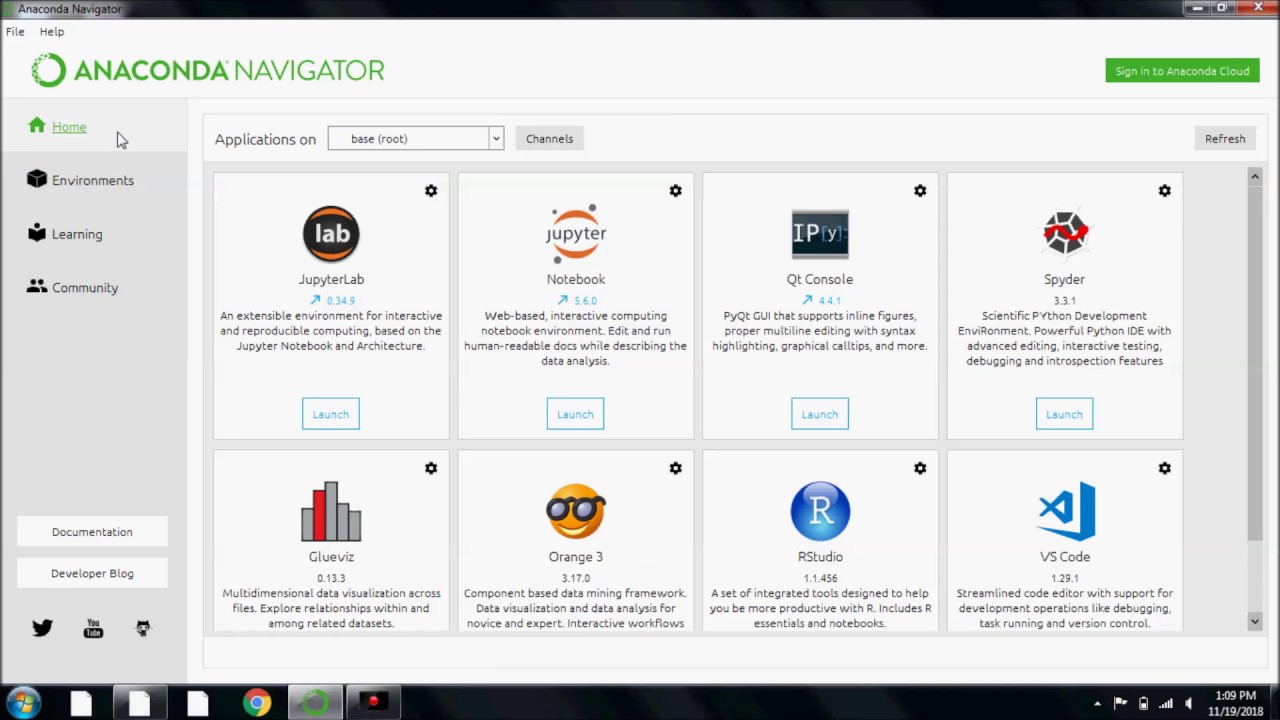
Anaconda Navigator is a desktop [graphical user interface (GUI)](https://en.wikipedia.org/wiki/Graphical_user_interface) included in Anaconda distribution that allows users to launch applications and manage conda packages, environments and channels without using [command-line commands](https://en.wikipedia.org/wiki/Command-line_interface). Navigator can search for packages on Anaconda Cloud or in a local Anaconda Repository, install them in an environment, run the packages and update them. It is available for [Windows](https://en.wikipedia.org/wiki/Windows), [macOS](https://en.wikipedia.org/wiki/MacOS) and [Linux](https://en.wikipedia.org/wiki/Linux).

## Why use Navigator?

In order to run, many scientific packages depend on specific versions of other packages. Data scientists often use multiple versions of many packages, and use multiple environments to separate these different versions.

The command line program conda is both a package manager and an environment manager, to help data scientists ensure that each version of each package has all the dependencies it requires and works correctly.

Navigator is an easy, point-and-click way to work with packages and environments without needing to type conda commands in a terminal window. You can use it to find the packages you want, install them in an environment, run the packages and update them, all inside Navigator.



**Conda**

Conda is an [open source](https://en.wikipedia.org/wiki/Open-source_software),[cross-platform](https://en.wikipedia.org/wiki/Cross-platform),language-agnostic [package manager](https://en.wikipedia.org/wiki/Package_manager) and environment management system that installs, runs, and updates packages and their dependencies. It was created for Python programs, but it can package and distribute software for any language (e.g., [R](https://en.wikipedia.org/wiki/R_(programming_language))), including multi-language projects. The Conda package and environment manager is included in all versions of Anaconda, Miniconda, and Anaconda Repository.

## Anaconda Cloud

Anaconda Cloud is a package management service by Anaconda where you can find, access, store and share public and private notebooks, environments, and conda and PyPI packages. Cloud hosts useful Python packages, notebooks and environments for a wide variety of applications. You do not need to log in or to have a Cloud account, to search for public packages, download and install them.

You can build new packages using the Anaconda Client command line interface (CLI), then manually or automatically upload the packages to Cloud.

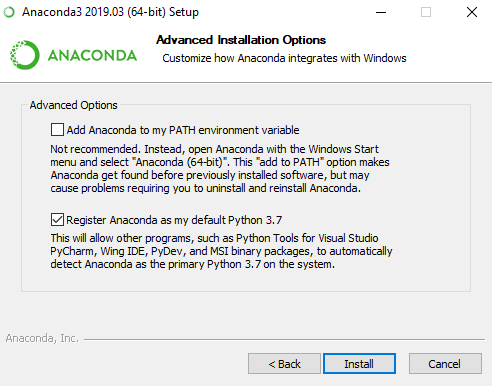
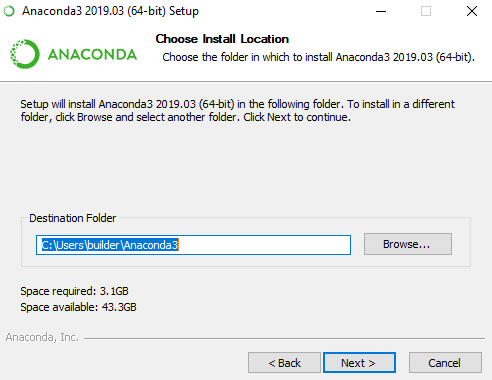
## What applications can I access using Navigator?

The following applications are available by default in Navigator:

* [JupyterLab](https://jupyterlab.readthedocs.io/en/stable/)
* [Jupyter Notebook](https://jupyter.readthedocs.io/en/latest/)
* [QTConsole](https://qtconsole.readthedocs.io/en/latest/)
* [Spyder](https://www.spyder-ide.org/)
* [VSCode](https://code.visualstudio.com/docs)
* [Glueviz](http://glueviz.org/en/stable/)
* [Orange 3 App](http://orange.biolab.si/docs/)
* [Rodeo](http://rodeo.yhat.com/docs/#requirements)
* [RStudio](http://docs.rstudio.com/)

# **Installing on Windows**

1. [Download the Anaconda installer](https://www.anaconda.com/download/#windows).
2. Double click the installer to launch.
3. Click Next.
4. Read the licensing terms and click “I Agree”.
5. Select an install for “Just Me” unless you’re installing for all users (which requires Windows Administrator privileges) and click Next.
6. Select a destination folder to install Anaconda and click the Next button.

[](https://docs.anaconda.com/_images/win-install-options.png)

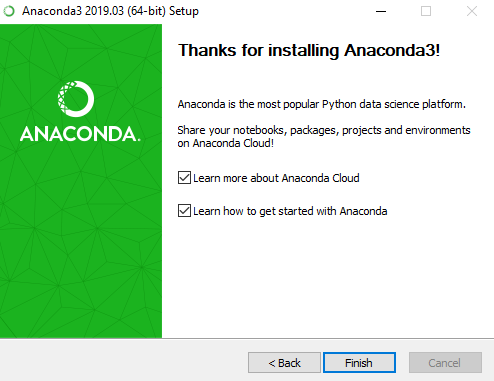
7.Choose whether to add Anaconda to your PATH environment variable. We recommend not adding Anaconda to the PATH environment variable, since this can interfere with other software. Instead, use Anaconda software by opening Anaconda Navigator or the Anaconda Prompt from the Start Menu.

8. Choose whether to register Anaconda as your default Python. Unless you plan on installing and running multiple versions of Anaconda, or multiple versions of Python, accept the default and leave this box checked.

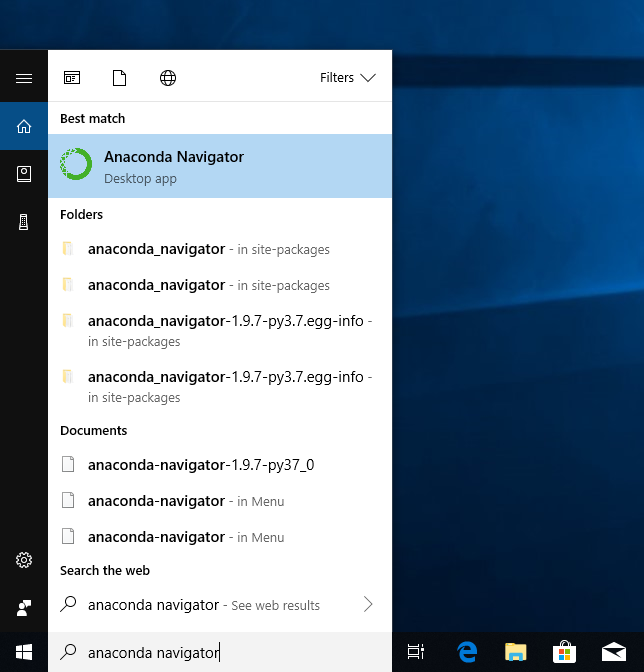
9. Click the Install button. If you want to watch the packages Anaconda is installing, click Show Details.

10. Click the Next button.

11. After a successful installation you will see the “Thanks for installing Anaconda” dialog box:

[](https://docs.anaconda.com/_images/win-install-complete.png)

12. After your install is complete, verify it by opening Anaconda Navigator, a program that is included with Anaconda: from your Windows Start menu, select the shortcut Anaconda Navigator from the Recently added or by typing “Anaconda Navigator”. If Navigator opens, you have successfully installed Anaconda. If not, check that you completed each step above

[](https://docs.anaconda.com/_images/win-navigator.png)

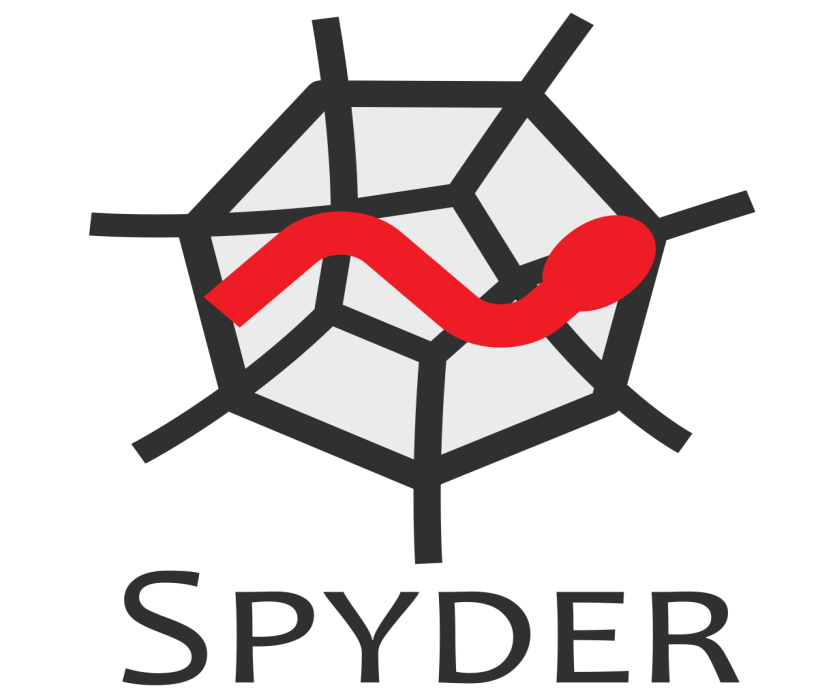
**SPYDER**

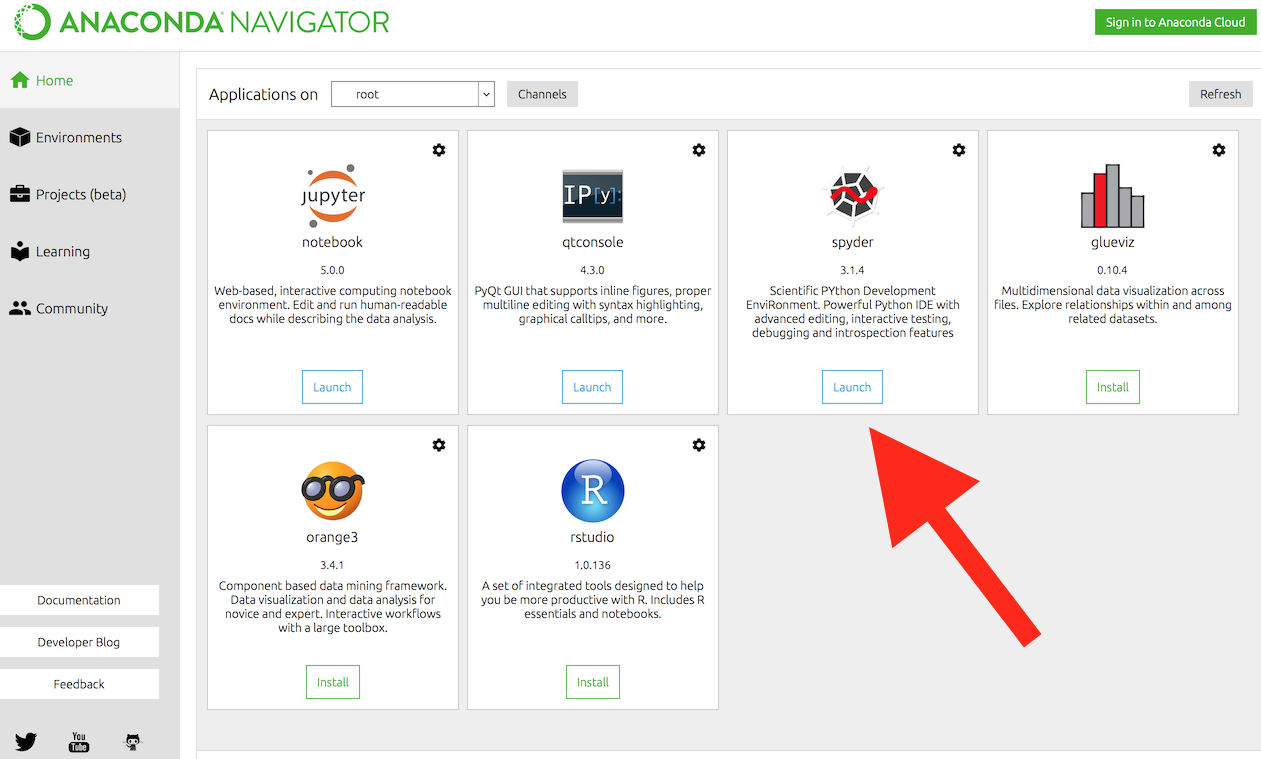
Spyder is a powerful scientific environment written in Python, for Python, and designed by and for scientists, engineers and data analysts. It offers a unique combination of the advanced editing, analysis, debugging, and profiling functionality of a comprehensive development tool with the data exploration, interactive execution, deep inspection, and beautiful visualization capabilities of a scientific package.

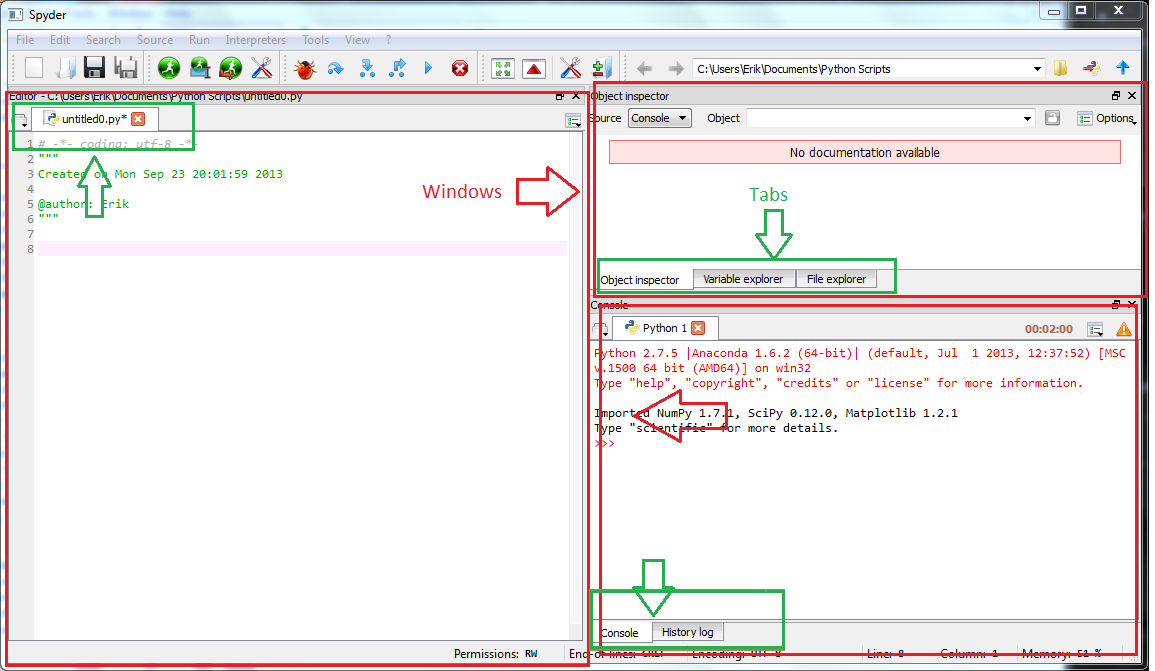
Spyder is an [open source](https://en.wikipedia.org/wiki/Open-source_software) cross-platform [integrated development environment](https://en.wikipedia.org/wiki/Integrated_development_environment) (IDE) for scientific programming in the [Python language](https://en.wikipedia.org/wiki/Python_(programming_language)). Spyder integrates with a number of prominent packages in the scientific Python stack, including [NumPy](https://en.wikipedia.org/wiki/NumPy), [SciPy](https://en.wikipedia.org/wiki/SciPy), [Matplotlib](https://en.wikipedia.org/wiki/Matplotlib), [pandas](https://en.wikipedia.org/wiki/Pandas_(software)), [IPython](https://en.wikipedia.org/wiki/IPython), [SymPy](https://en.wikipedia.org/wiki/SymPy) and [Cython](https://en.wikipedia.org/wiki/Cython), as well as other open source software. It is released under the [MIT license](https://en.wikipedia.org/wiki/MIT_license). Initially created and developed by Pierre Raybaut in 2009, since 2012 Spyder has been maintained and continuously improved by a team of scientific Python developers and the community.

Spyder is extensible with first- and third-party plugins,includes support for interactive tools for data inspection and embeds Python-specific code quality assurance and introspection instruments, such as Pyflakes, [Pylint](https://en.wikipedia.org/wiki/Pylint" \o "Pylint) and Rope. It is available cross-platform through [Anaconda](https://en.wikipedia.org/wiki/Anaconda_(Python_distribution)), on Windows, on macOS through [MacPorts](https://en.wikipedia.org/wiki/MacPorts" \o "MacPorts), and on major Linux distributions such as [Arch Linux](https://en.wikipedia.org/wiki/Arch_Linux), [Debian](https://en.wikipedia.org/wiki/Debian), [Fedora](https://en.wikipedia.org/wiki/Fedora_(operating_system)), [Gentoo Linux](https://en.wikipedia.org/wiki/Gentoo_Linux), [openSUSE](https://en.wikipedia.org/wiki/OpenSUSE) and [Ubuntu](https://en.wikipedia.org/wiki/Ubuntu_(operating_system)).

|  |  |
| --- | --- |
| **Written in** | [Python](https://en.wikipedia.org/wiki/Python_(programming_language)) |



****



**Features Of Spyder**

* An editor with [syntax highlighting](https://en.wikipedia.org/wiki/Syntax_highlighting), [introspection](https://en.wikipedia.org/wiki/Introspection_(computer_science)), [code completion](https://en.wikipedia.org/wiki/Code_completion).
* Support for multiple [IPython](https://en.wikipedia.org/wiki/IPython" \o "IPython) [consoles](https://en.wikipedia.org/wiki/Command-line_interface).
* The ability to explore and edit [variables](https://en.wikipedia.org/wiki/Variable_(computer_science)) from a [GUI](https://en.wikipedia.org/wiki/Graphical_user_interface).
* A Help pane able to retrieve and render rich text [documentation](https://en.wikipedia.org/wiki/Application_programming_interface#Documentation) on functions, classes and methods automatically or on-demand.
* A [debugger](https://en.wikipedia.org/wiki/Debugger) linked to IPdb, for step-by-step execution.
* [Static code analysis](https://en.wikipedia.org/wiki/Static_program_analysis), powered by [Pylint](https://en.wikipedia.org/wiki/Pylint" \o "Pylint).
* A run-time [Profiler](https://en.wikipedia.org/wiki/Profiling_(computer_programming)), to benchmark code.
* Project support, allowing work on multiple development efforts simultaneously.
* A built-in [file explorer](https://en.wikipedia.org/wiki/File_manager), for interacting with the filesystem and managing projects.
* A "Find in Files" feature, allowing full [regular expression](https://en.wikipedia.org/wiki/Regular_expression) search over a specified scope.
* An online help browser, allowing users to search and view Python and package documentation inside the IDE.
* A [history log](https://en.wikipedia.org/wiki/Command_history), recording every user command entered in each console.

**LIBRARIES**

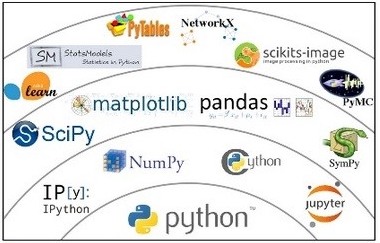
It is collection of **modules.**(Library either contains built in modules(written in C) + modules written in python).

Python library is a collection of functions and methods that allows you to perform lots of actions without writing your own code. For example, if you are working with data, **numpy**, **scipy**, **pandas**, etc.

The Python Standard Library is a collection of script modules accessible to a Python program to simplify the programming process and removing the need to rewrite commonly used commands. They can be used by 'calling/importing' them at the beginning of a script.

A list of the Standard Library modules can be found at <http://www.python.org/doc/>

A Python library is a reusable chunk of code that you may want to include in your programs/ projects. Compared to languages like C++ or C, a Python libraries do not pertain to any specific context in Python. Here, a ‘library’ loosely describes a collection of core modules. Essentially, then, a library is a collection of modules. A package is a library that can be installed using a package manager like rubygems or npm.

The Python Standard Library is a collection of exact syntax, token, and semantics of Python. It comes bundled with core Python distribution. We mentioned this when we began with an introduction.

**ARCHITECTURE:**

## **Introduction to Game**

Games are about moving pixels on the screen and making noise. Pretty much all video/computer games have most of the following elements.

### Main Loop

The main loop of a game runs and refreshes the screen at fixed intervals. This is your frame rate, and it dictates how smooth things are. Typically, games refresh the screen 30 to 60 times a second. If you go slower, objects on the screen will seem jerky.

Inside the main loop, there are three main activities: handling events, updating the game state, and drawing the current state of the screen.

### Handling Events

Events in a game consist of everything that happens outside the control of the game's code but is relevant to the operation of the game. For example, if in Breakout the player presses the left arrow key, the game needs to move the paddle to the left. Typical events are key presses (and releases), mouse movement, mouse button clicks (especially in menus), and timer events (e.g. a special effect expires after 10 seconds).

### Drawing

The game needs to display its state on the screen. This includes drawing geometrical shapes, images, and text.

### Game Physics

Most games simulate a physical environment. In Breakout, the ball bounces off objects and has a very crude rigid-body physics system in place (if you can call it that).

More advanced games may have more sophisticated and realistic physics systems (especially 3D games). Note that some games like card games don't have much physics at all, and that's totally fine.

### Lives, Score, and Levels

Most games give you a certain amount of lives, and when you run out of lives, the game is over. You also often have a score that gives you a sense of how well you're doing and a motivation to improve next time you play or just brag to your friends about your Breakout mad skills. Many games have levels that are either completely different or raise the level of difficulty.

### What's Pygame?

Pygame is a Python framework for game programming. It is built on top of [SDL](https://en.wikipedia.org/wiki/Simple_DirectMedia_Layer) and has all the good stuff:

* mature
* great community
* open source
* cross-platform
* good docs
* plenty of sample games
* easy to learn

### Installing Pygame

Type pip install pygame in order to install it. If you need something else, follow the instructions in the [Getting Started](https://www.pygame.org/wiki/GettingStarted) section of the Wiki. If you run macOS Sierra as I do, you may run into some trouble. I was able to install Pygame with no trouble, and the code seemed to run just fine, but the game window never showed up.

That's kind of a bummer when you run a game. I eventually had to resort to running on Windows in a VirtualBox VM. Hopefully, by the time you read this article, the issue will have been resolved.

# **Program Designing:**

# **Normalization**

Most games now have an online element to them; allowing users to take part in leaderboards, join group games or chat to others. Internet connectivity in a game adds a new opportunity for gamers as it allows players to find and play against, or with, other players. These may be their friends or family members or even other users in the game from around the world (in a multi-player game). User account is required to play games online and store the progress of user. All these data is saved in the database of the online gaming platform.

* The process of organizing the data in the database is called **normalization**.
* Normalization is used to minimize the redundancy from a relation or set of relations. It is also used to eliminate the undesirable characteristics like Insertion, Update and Deletion Anomalies.
* Normalization divides the larger table into the smaller table and links them using relationship.
* The normal form is used to reduce redundancy from the database table.

## **Types of Normal Forms**

There are the four types of normal forms:

|  |  |
| --- | --- |
| **Normal Form** | **Description** |
| [1NF](https://www.javatpoint.com/dbms-first-normal-form) | A relation is in 1NF if it contains an atomic value. |
| [2NF](https://www.javatpoint.com/dbms-second-normal-form) | A relation will be in 2NF if it is in 1NF and all non-key attributes are fully functional dependent on the primary key. |
| [3NF](https://www.javatpoint.com/dbms-third-normal-form) | A relation will be in 3NF if it is in 2NF and no transition dependency exists. |
| [4NF](https://www.javatpoint.com/dbms-forth-normal-form) | A relation will be in 4NF if it is in Boyce Codd normal form and has no multi-valued dependency. |
| [5NF](https://www.javatpoint.com/dbms-fifth-normal-form) | A relation is in 5NF if it is in 4NF and not contains any join dependency and joining should be lossless. |

# **Game Testing:**

**Game testing**, a subset of game development, is a software testing process for quality control of video games. The primary function of game testing is the discovery and documentation of software defects. Interactive entertainment software testing is a highly technical field requiring computing expertise, analytic competence, critical evaluation skills, and endurance. In recent years the field of game testing has come under fire for being extremely strenuous and unrewarding, both financially and emotionally.

Modern games are some of the most complex software expressions in existence, so you cannot simply use existing software testing processes to test them. Game testing nowadays requires the right understanding of the gaming industry, title publishing, mass server loading, level architecture and advanced user experience. It can often entail testing rarely used in other software scenarios such as AI testing, multiplayer networking, audio testing, Physics and realism, and API integration.

**Game Testing Processes**

The gaming industry is, so far, the most rapidly expanding software sector. As such, the sophistication and complexity of modern titles bring with them a diverse range of features (both software and hardware), UI, functionality due to interaction, server concerns with online multiplayer options and security regarding accounts and payments.

 Here are the most common game testing processes:

* + - * 1. Alpha/Beta Testing
        2. Feature Testing
        3. Smoke Testing
        4. Automated Testing
        5. Localization Testing

**Game Testing Phases**

Below are the game testing phases, which are somewhat similar to the software testing life cycle (STLC):

**Gathering of Requirements**

To create an effective game testing strategy, testers should have a thorough understanding of the following aspects of the game:

• Storyboard

• Architecture

• Features

• Characters

• Concept

• Rules

• Pointing System

• Stages/Levels

Without thorough knowledge of the following requirements and the software testing process, then it would be possible for testers to miss defects or other more errors which can affect the final product.

**Preparation of Game Test Strategy**

In this phase, the Test Strategy document is prepared, which must include sections like:

• Timeline

• Testers

• Number of testing cycles

• In-scope

• Out-scope

• Testing types to be conducted

• Risk-based test analysis

• Service Level Agreements (SLAs)

• Risks and mitigation

• Defect logging process

• Reporting process

Game developers and testers both should be well familiar with the software testing process and development process and strategy of the game development phase and the important specifics that are included during each stage.

**To Do’s When Testing Games**

Games need to be tested before, during, and after their release in the market. Once released, gamers expect a near flawless experience, and it is game testers’ job to ensure that gamers’ expectation will be met, even surpassed.

Here are some tips for testing game titles:

1. **Review the game rules.**
2. **Verify functionality.**
3. **Check for compliance.**
4. **Make sure it is localized.**
5. **Do the game soaking.**

Game soaking is done by exposing a game for an excessively long period of execution. The purpose of this activity is to test if the game will display unreleased segments of memory and mathematical rounding discrepancies. These errors can cause problems as they can affect game parameters involving numerical computations, such as pointing system and health value consumption.

**Perform security test**

## Key Risks in Game Testing

1. The game does not create compelling experiences for the targeted audience.
2. The game does not have a player-centric design
3. The fun factor and addictive gameplay missing in the games.
4. Game not unique, competitive, fast paced.
5. The game fails because of technical issues, broken features, critical bugs, bad music sound, and poor video.
6. Game development cost goes over budget
7. The game should have simple aesthetic design and the gameplay.

**Program:**

"""

Code for **Pylock game**

"""

import pygame

import random

"""

10 x 20 square grid

shapes: S, Z, I, O, J, L, T

represented in order by 0 - 6

"""

pygame.font.init()

**# GLOBALS VARS**

s\_width = 800

s\_height = 700

play\_width = 300 # meaning 300 // 10 = 30 width per block

play\_height = 600 # meaning 600 // 20 = 20 height per blo ck

block\_size = 30

top\_left\_x = (s\_width - play\_width) // 2

top\_left\_y = s\_height - play\_height

**# SHAPE FORMATS**

S = [['.....',

'.....',

'..00.',

'.00..',

'.....'],

['.....',

'..0..',

'..00.',

'...0.',

'.....']]

Z = [['.....',

'.....',

'.00..',

'..00.',

'.....'],

['.....',

'..0..',

'.00..',

'.0...',

'.....']]

I = [['..0..',

'..0..',

'..0..',

'..0..',

'.....'],

['.....',

'0000.',

'.....',

'.....',

'.....']]

O = [['.....',

'.....',

'.00..',

'.00..',

'.....']]

J = [['.....',

'.0...',

'.000.',

'.....',

'.....'],

['.....',

'..00.',

'..0..',

'..0..',

'.....'],

['.....',

'.....',

'.000.',

'...0.',

'.....'],

['.....',

'..0..',

'..0..',

'.00..',

'.....']]

L = [['.....',

'...0.',

'.000.',

'.....',

'.....'],

['.....',

'..0..',

'..0..',

'..00.',

'.....'],

['.....',

'.....',

'.000.',

'.0...',

'.....'],

['.....',

'.00..',

'..0..',

'..0..',

'.....']]

T = [['.....',

'..0..',

'.000.',

'.....',

'.....'],

['.....',

'..0..',

'..00.',

'..0..',

'.....'],

['.....',

'.....',

'.000.',

'..0..',

'.....'],

['.....',

'..0..',

'.00..',

'..0..',

'.....']]

shapes = [S, Z, I, O, J, L, T]

shape\_colors = [(0, 255, 0), (255, 0, 0), (0, 255, 255), (255, 255, 0), (255, 165, 0), (0, 0, 255), (128, 0, 128)]

# index 0 - 6 represent shape

class Piece(object):

rows = 20 # y

columns = 10 # x

def \_\_init\_\_(self, column, row, shape):

self.x = column

self.y = row

self.shape = shape

self.color = shape\_colors[shapes.index(shape)]

self.rotation = 0 # number from 0-3

def create\_grid(locked\_positions={}):

grid = [[(0,0,0) for x in range(10)] for x in range(20)]

for i in range(len(grid)):

for j in range(len(grid[i])):

if (j,i) in locked\_positions:

c = locked\_positions[(j,i)]

grid[i][j] = c

return grid

def convert\_shape\_format(shape):

positions = []

format = shape.shape[shape.rotation % len(shape.shape)]

for i, line in enumerate(format):

row = list(line)

for j, column in enumerate(row):

if column == '0':

positions.append((shape.x + j, shape.y + i))

for i, pos in enumerate(positions):

positions[i] = (pos[0] - 2, pos[1] - 4)

return positions

def valid\_space(shape, grid):

accepted\_positions = [[(j, i) for j in range(10) if grid[i][j] == (0,0,0)] for i in range(20)]

accepted\_positions = [j for sub in accepted\_positions for j in sub]

formatted = convert\_shape\_format(shape)

for pos in formatted:

if pos not in accepted\_positions:

if pos[1] > -1:

return False

return True

def check\_lost(positions):

for pos in positions:

x, y = pos

if y < 1:

return True

return False

def get\_shape():

global shapes, shape\_colors

return Piece(5, 0, random.choice(shapes))

def draw\_text\_middle(text, size, color, surface):

font = pygame.font.SysFont('comicsans', size, bold=True)

label = font.render(text, 1, color)

surface.blit(label, (top\_left\_x + play\_width/2 - (label.get\_width() / 2), top\_left\_y + play\_height/2 - label.get\_height()/2))

def draw\_grid(surface, row, col):

sx = top\_left\_x

sy = top\_left\_y

for i in range(row):

pygame.draw.line(surface, (128,128,128), (sx, sy+ i\*30), (sx + play\_width, sy + i \* 30)) # horizontal lines

for j in range(col):

pygame.draw.line(surface, (128,128,128), (sx + j \* 30, sy), (sx + j \* 30, sy + play\_height)) # vertical lines

def clear\_rows(grid, locked):

# need to see if row is clear the shift every other row above down one

inc = 0

for i in range(len(grid)-1,-1,-1):

row = grid[i]

if (0, 0, 0) not in row:

inc += 1

# add positions to remove from locked

ind = i

for j in range(len(row)):

try:

del locked[(j, i)]

except:

continue

if inc > 0:

for key in sorted(list(locked), key=lambda x: x[1])[::-1]:

x, y = key

if y < ind:

newKey = (x, y + inc)

locked[newKey] = locked.pop(key)

def draw\_next\_shape(shape, surface):

font = pygame.font.SysFont('comicsans', 30)

label = font.render('Next Shape', 1, (255,255,255))

sx = top\_left\_x + play\_width + 50

sy = top\_left\_y + play\_height/2 - 100

format = shape.shape[shape.rotation % len(shape.shape)]

for i, line in enumerate(format):

row = list(line)

for j, column in enumerate(row):

if column == '0':

pygame.draw.rect(surface, shape.color, (sx + j\*30, sy + i\*30, 30, 30), 0)

surface.blit(label, (sx + 10, sy- 30))

def draw\_window(surface):

surface.fill((0,0,0))

# Tetris Title

font = pygame.font.SysFont('comicsans', 60)

label = font.render('TETRIS', 1, (255,255,255))

surface.blit(label, (top\_left\_x + play\_width / 2 - (label.get\_width() / 2), 30))

for i in range(len(grid)):

for j in range(len(grid[i])):

pygame.draw.rect(surface, grid[i][j], (top\_left\_x + j\* 30, top\_left\_y + i \* 30, 30, 30), 0)

# draw grid and border

draw\_grid(surface, 20, 10)

pygame.draw.rect(surface, (255, 0, 0), (top\_left\_x, top\_left\_y, play\_width, play\_height), 5)

# pygame.display.update()

def main():

global grid

locked\_positions = {} # (x,y):(255,0,0)

grid = create\_grid(locked\_positions)

change\_piece = False

run = True

current\_piece = get\_shape()

next\_piece = get\_shape()

clock = pygame.time.Clock()

fall\_time = 0

level\_time = 0

fall\_speed = 0.27

score = 0

while run:

grid = create\_grid(locked\_positions)

fall\_time += clock.get\_rawtime()

level\_time += clock.get\_rawtime()

clock.tick()

if level\_time/1000 > 4:

level\_time = 0

if fall\_speed > 0.15:

fall\_speed -= 0.005

**# PIECE FALLING CODE**

if fall\_time/1000 >= fall\_speed:

fall\_time = 0

current\_piece.y += 1

if not (valid\_space(current\_piece, grid)) and current\_piece.y > 0:

current\_piece.y -= 1

change\_piece = True

for event in pygame.event.get():

if event.type == pygame.QUIT:

run = False

pygame.display.quit()

quit()

if event.type == pygame.KEYDOWN:

if event.key == pygame.K\_LEFT:

current\_piece.x -= 1

if not valid\_space(current\_piece, grid):

current\_piece.x += 1

elif event.key == pygame.K\_RIGHT:

current\_piece.x += 1

if not valid\_space(current\_piece, grid):

current\_piece.x -= 1

elif event.key == pygame.K\_UP:

# rotate shape

current\_piece.rotation = current\_piece.rotation + 1 % len(current\_piece.shape)

if not valid\_space(current\_piece, grid):

current\_piece.rotation = current\_piece.rotation - 1 % len(current\_piece.shape)

if event.key == pygame.K\_DOWN:

# move shape down

current\_piece.y += 1

if not valid\_space(current\_piece, grid):

current\_piece.y -= 1

shape\_pos = convert\_shape\_format(current\_piece)

# add piece to the grid for drawing

for i in range(len(shape\_pos)):

x, y = shape\_pos[i]

if y > -1:

grid[y][x] = current\_piece.color

**# IF PIECE HIT GROUND**

if change\_piece:

for pos in shape\_pos:

p = (pos[0], pos[1])

locked\_positions[p] = current\_piece.color

current\_piece = next\_piece

next\_piece = get\_shape()

change\_piece = False

# call four times to check for multiple clear rows

if clear\_rows(grid, locked\_positions):

score += 10

draw\_window(win)

draw\_next\_shape(next\_piece, win)

pygame.display.update()

# Check if user lost

if check\_lost(locked\_positions):

run = False

draw\_text\_middle("You Lost", 40, (255,255,255), win)

pygame.display.update()

pygame.time.delay(2000)

def main\_menu():

run = True

while run:

win.fill((0,0,0))

draw\_text\_middle('Press any key to begin.', 60, (255, 255, 255), win)

pygame.display.update()

for event in pygame.event.get():

if event.type == pygame.QUIT:

run = False

if event.type == pygame.KEYDOWN:

main()

pygame.quit()

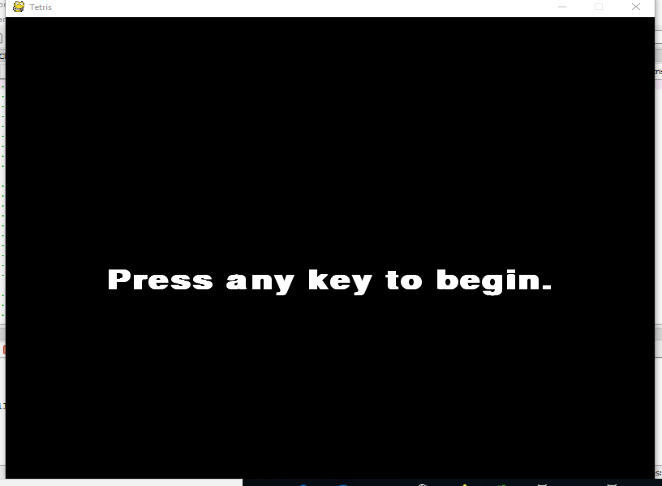
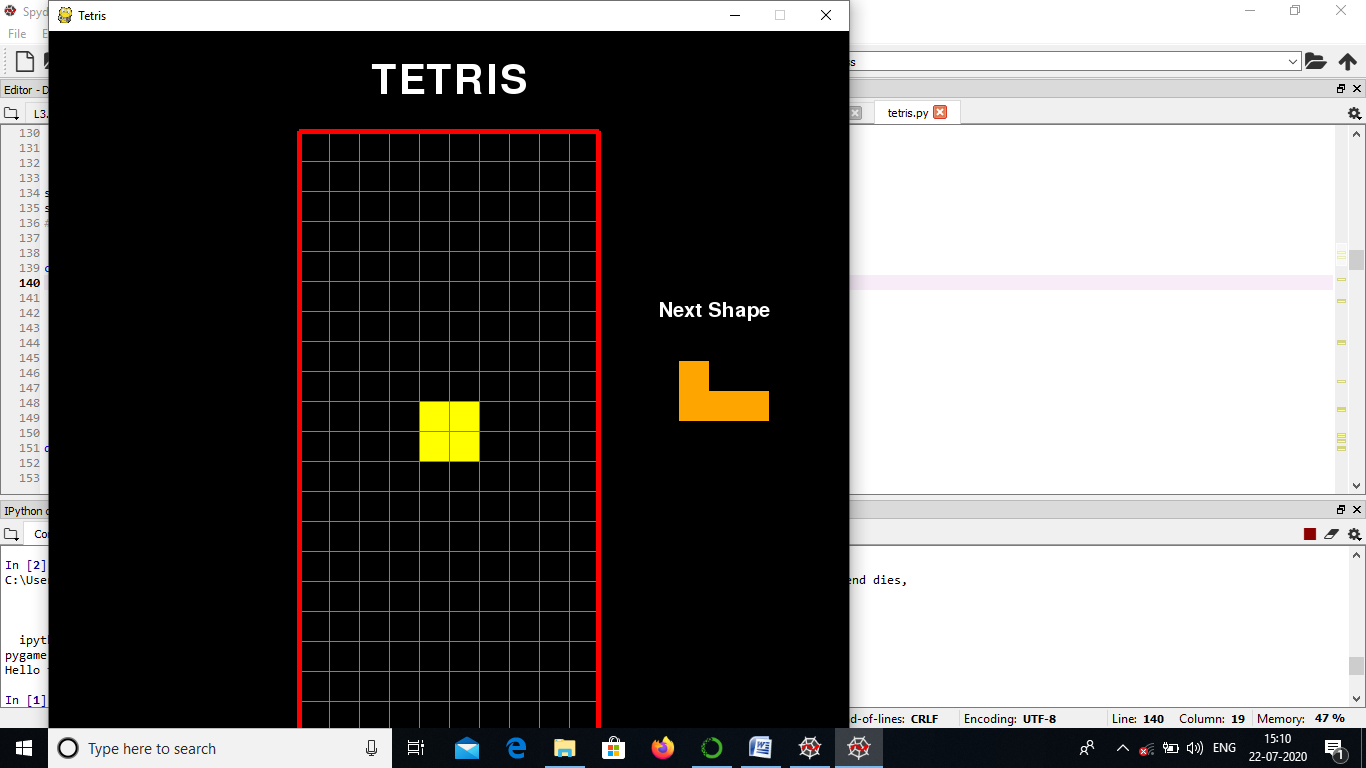
win = pygame.display.set\_mode((s\_width, s\_height))

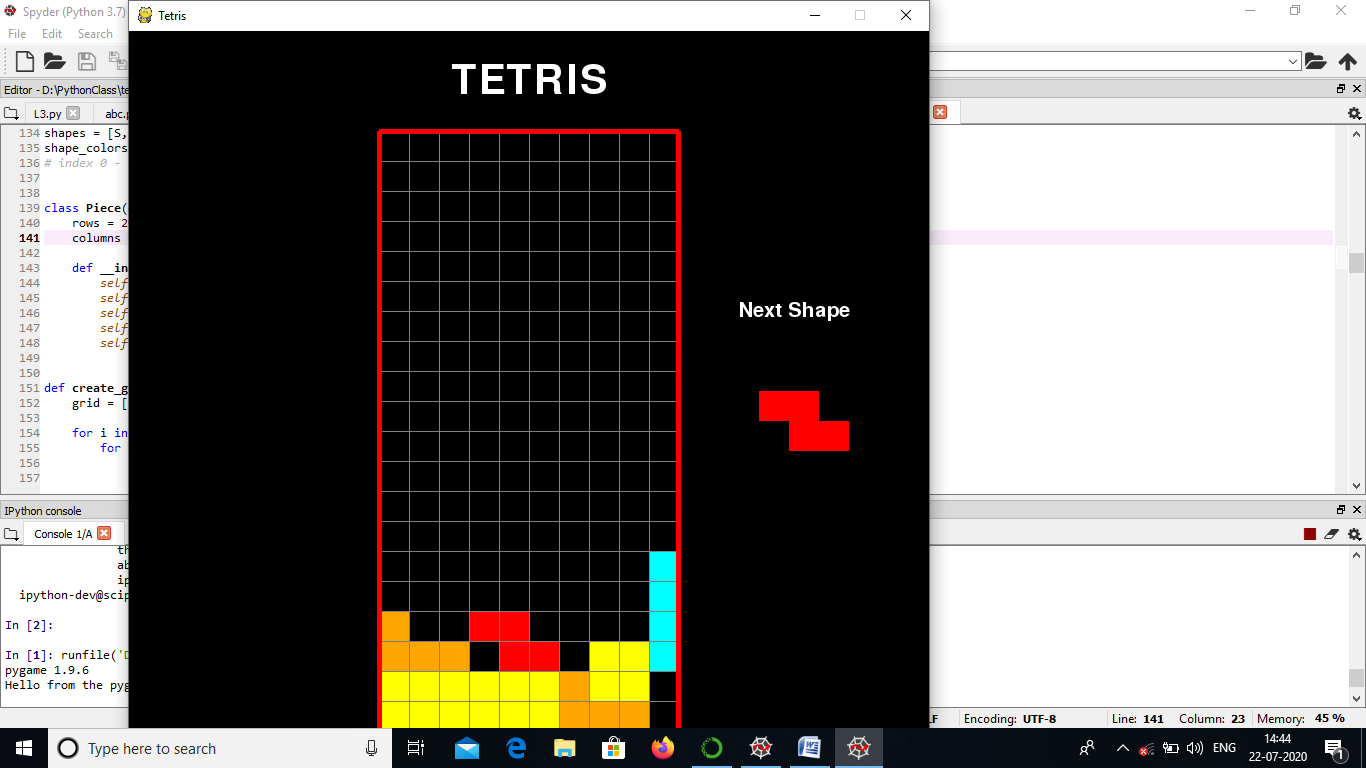
pygame.display.set\_caption('Tetris')

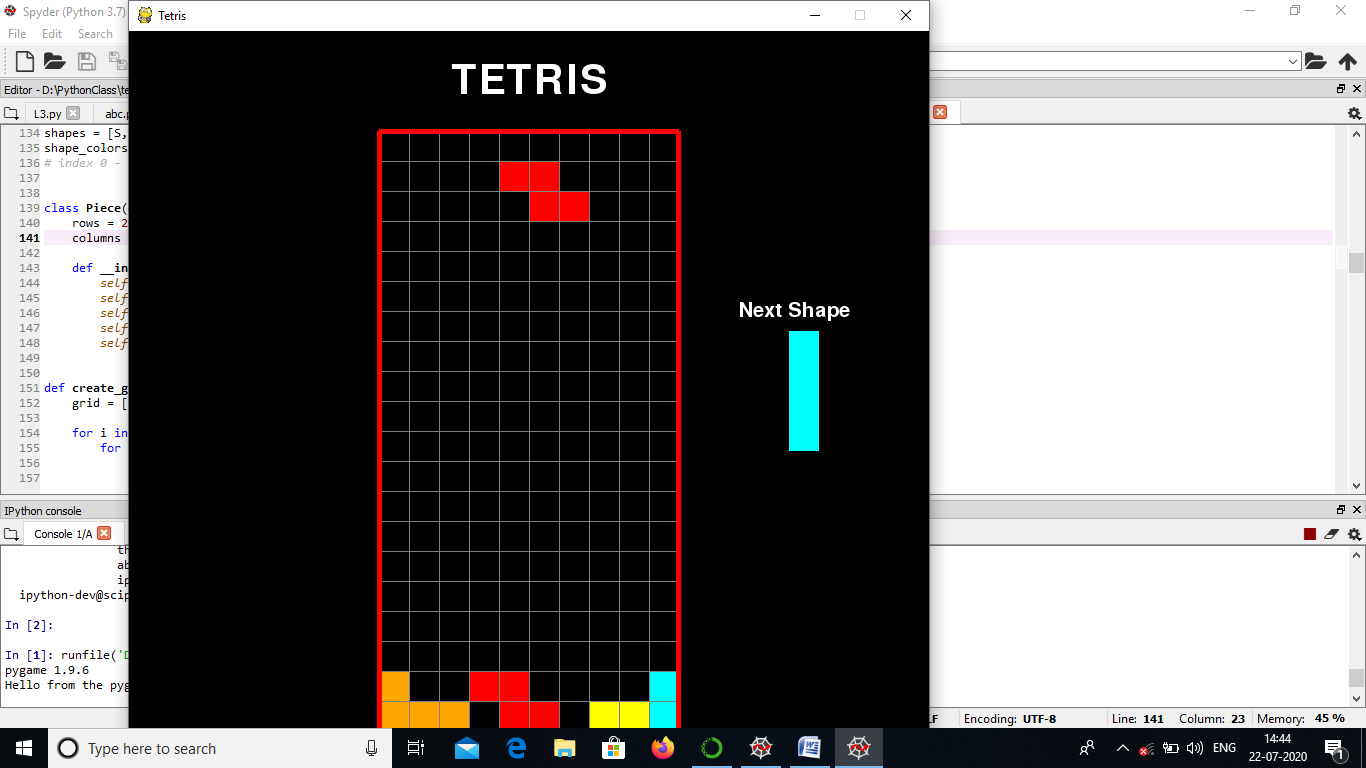
main\_menu()

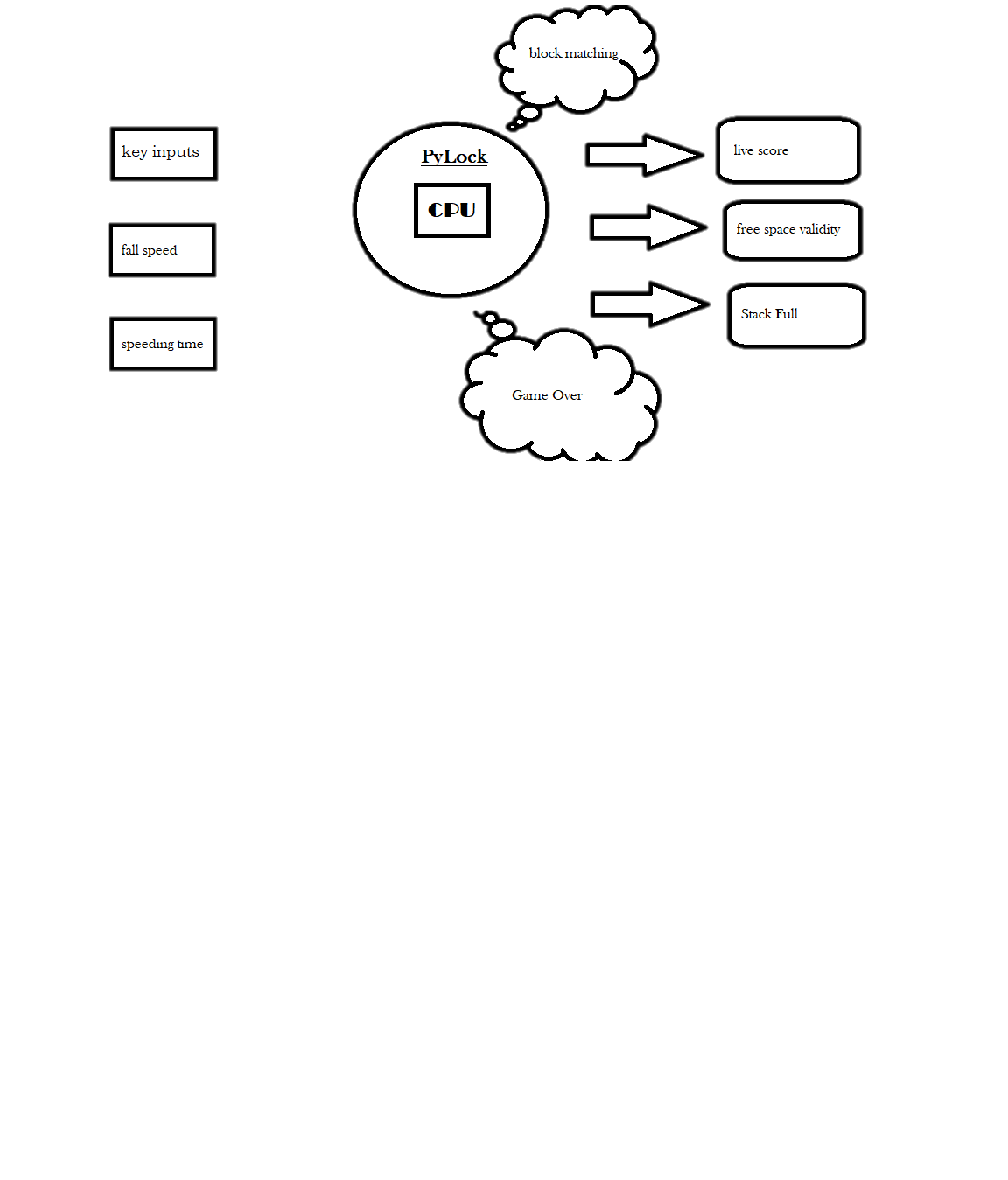
**# start game**

**Output:**

****

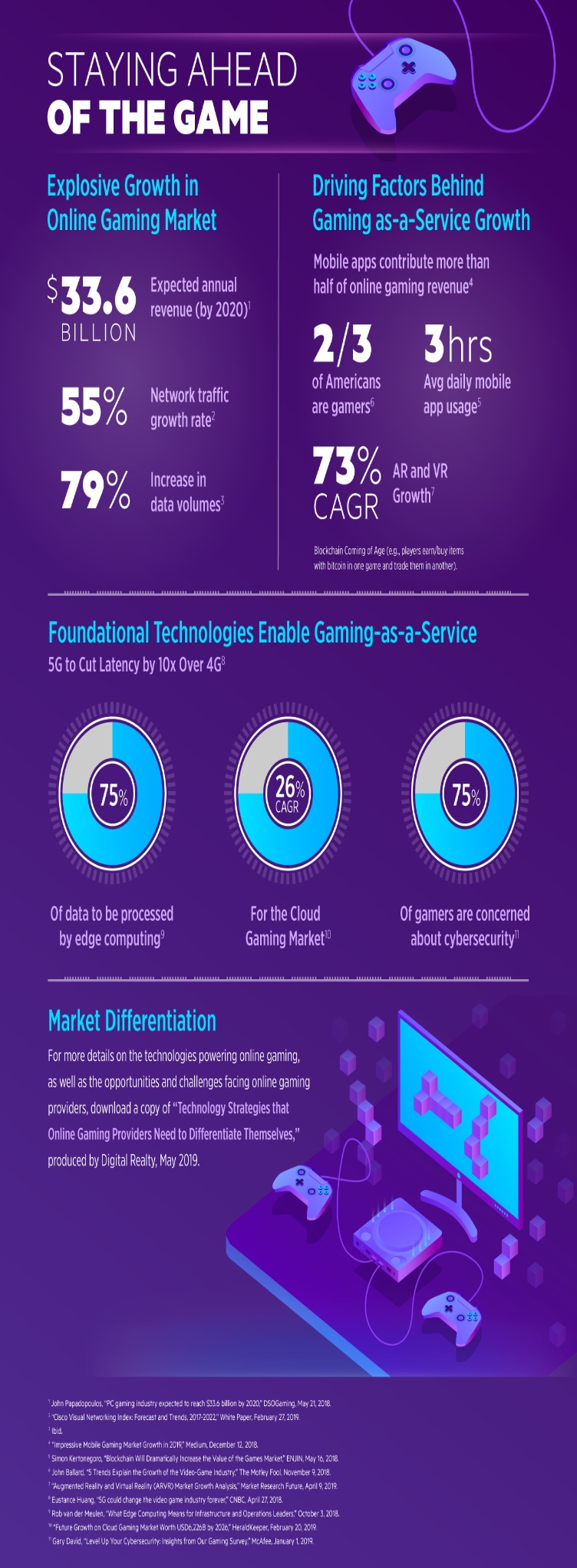
****

****

**Program Designing**:

**FUTURE SCOPE:**

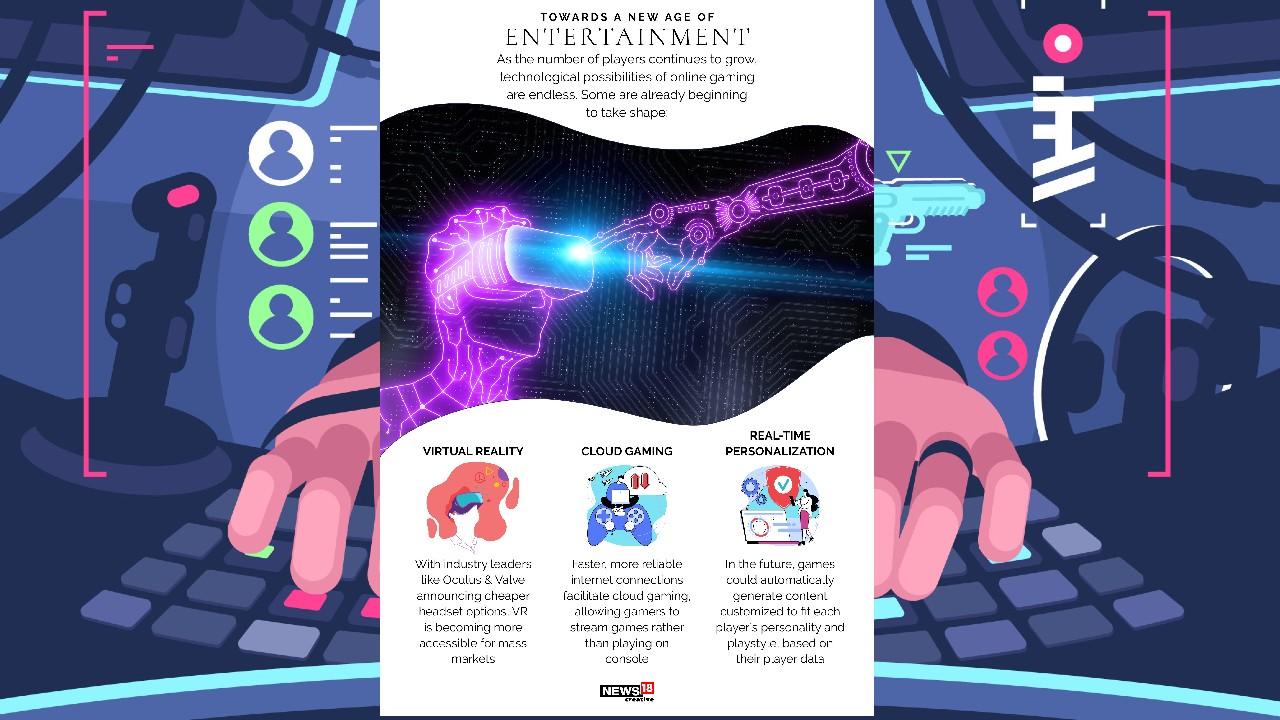
From board games to virtual reality, how we play games has changed with time. In just a few decades, the world of online gaming has exploded in popularity. The online gaming industry in India has entered a tremendous growth phase over recent years. Buoyed by smartphone and internet penetration, it has grown into a multi-million dollar industry in India. The online gaming is now considered to be one of the fastest growing industries on the planet. Here’s a deep dive into the humble beginnings of the online gaming and the technological possibilities driving its future.

As the number of players continues to grow technical possibilities of online gaming are endless. Some are already beginning to take shape like

With industry leaders like Oculus & Valve announcing cheaper headset options VR is becoming more accessible for mess markets.

Faster, more reliable internet connections facilitate cloud gaming, allowing games to stream games rather than playing on console.

In the future, games could automatically generate content customized to fit each player’s personality.



**Bibliography**

* <https://www.python.org/>
* <https://github.com>
* <https://www.testinghero.com/single-post/Beginners-Guide-to-Game-Testing>
* <https://www.javatpoint.com/dbms-normalization>
* <https://web.whatsapp.com/>
* <https://www.geeksforgeeks.org/>
* <https://www.tutorialspoint.com/index.htm>
* <https://www.researchgate>
* <https://stackoverflow.com>
* <https://www.anaconda.com/>