

Programming the Computer (Python)

Reinforcement Handout

Programming the Computer?

Computer programming is a way of giving computers instructions about what they should do next. These instructions are known as code, and computer programmers write code to solve problems or perform a task. The end goal is to create something: that could mean anything from a web page, or a piece of software, or even just a pretty picture. That's why computer programming is often described as a mix between art and science; it's technical and analytical, yet creative at the same time.

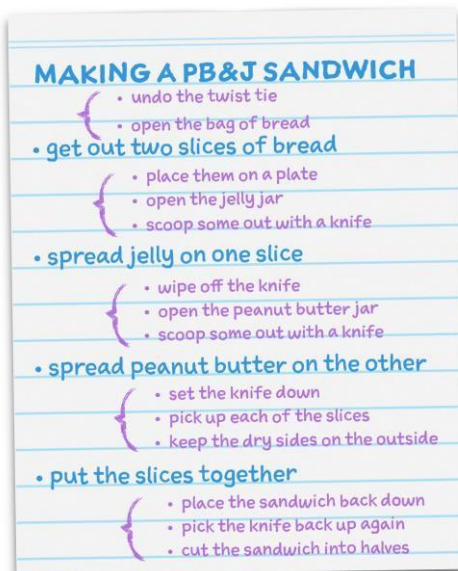
Flowchart and Algorithm:

Algorithms and flowcharts are two different tools used for creating new programs, especially in computer programming. An algorithm is a step-by-step analysis of the process, while a flowchart explains the steps of a program in a graphical way.

Algorithm:

To write a logical step-by-step method to solve the problem is called algorithm, in other words, an algorithm is a procedure for solving problems. In order to solve a mathematical or computer problem, this is the first step of the procedure. An algorithm includes calculations, reasoning and data processing. Algorithms can be presented by natural languages, pseudo code and flowcharts, etc.

Algorithm for making Peanut Butter Sandwich



Algorithm for taking valid user information

Step 1: Start

Step 2: Create a variable to receive the user's email address

Step 3: Clear the variable in case it's not empty

Step 4: Ask the user for an email address

Step 5: Store the response in the variable

Step 6: Check the stored response to see if it is a valid email address

Step 7: Not valid? Go back to Step 3.

Step 8: End

- Write an algorithm for making Pancake or any process related to your daily routine.
- Write an algorithm for a product purchase through online shopping.

Flowchart:

A flowchart is the graphical or pictorial representation of an algorithm with the help of different symbols, shapes and arrows in order to demonstrate a process or a program. With algorithms, we can easily understand a program. The main purpose of a flowchart is to analyze different processes. Several standard graphics are applied in a flowchart:

Basic Shapes used to make Flow Chart

Terminal Box -
Start / End



Input / output



**Process /
Instruction**



Decision



**Connector /
Arrow**



The graphics above represent different part of a flowchart. The process in a flowchart can be expressed through boxes and arrows with different sizes and colors. In a flowchart, we can easily highlight a certain element and the relationships between each part.

Reinforcement Handout: Programming Computer - Python

How to Use Flowcharts to Represent Algorithms:

Algorithms are mainly used for mathematical and computer programs, whilst flowcharts can be used to describe all sorts of processes: business, educational, personal and of course algorithms. So flowcharts are often used as a program planning tool to visually organize the step-by-step process of a program. Here are some examples:

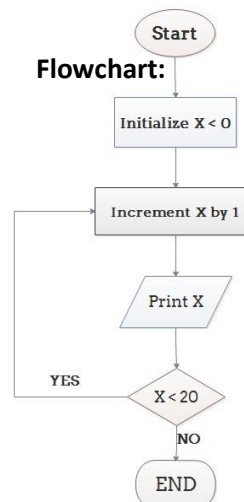
Example: Print 1 to 20:

Algorithm:

Step 1: Initialize X as 0, Step 2:

Increment X by 1, Step 3: Print X,

Step 4: If X is less than 20 then go back to step 2.

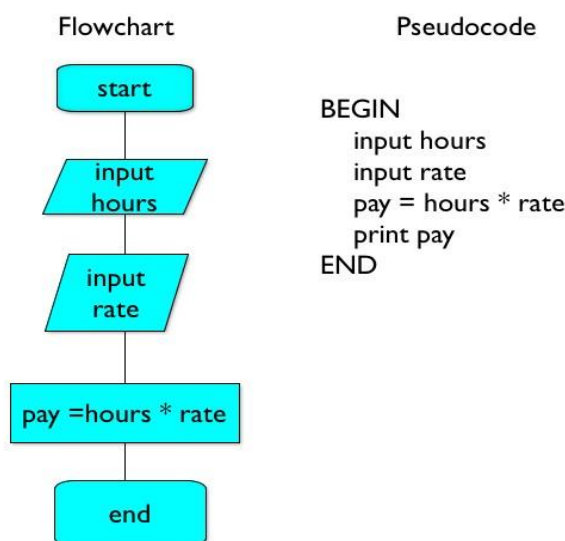


- Draw a flowchart using the algorithm for a product purchase through online shopping.

Pseudocode & Steps to Solution:

Pseudocode

Pseudocode is an informal high-level description of the operating principle of a computer program or other algorithm. It uses the structural conventions of a normal programming language, but is intended for human reading rather than machine reading. Pseudocode typically omits details that are essential for machine understanding of the algorithm, such as variable declarations, system-specific code etc. In simple words Pseudocode is a precise form of Algorithm without unnecessary steps of vocabulary/words.



- Convert the flowchart of making online shopping to pseudocode.

What is a programme?

In computing, a program is a specific set of ordered operations for a computer to perform. In the modern computer that John von Neumann outlined in 1945, the program contains a one-at-a-time sequence of instructions that the computer follows.

How Program interacts with Hardware?

Hardware is a term we use to describe the electronics and mechanical parts of the computer. To be able to use it, we need programs, the software. A computer program is a list of instructions stored as a file on a storage device. If this program is embedded inside a hardware device it is called firmware. When we run the program, the computer “reads” the list of commands or instructions and does what the program tells it to do.

Reinforcement Handout: Programming Computer - Python

Python Programming Language:

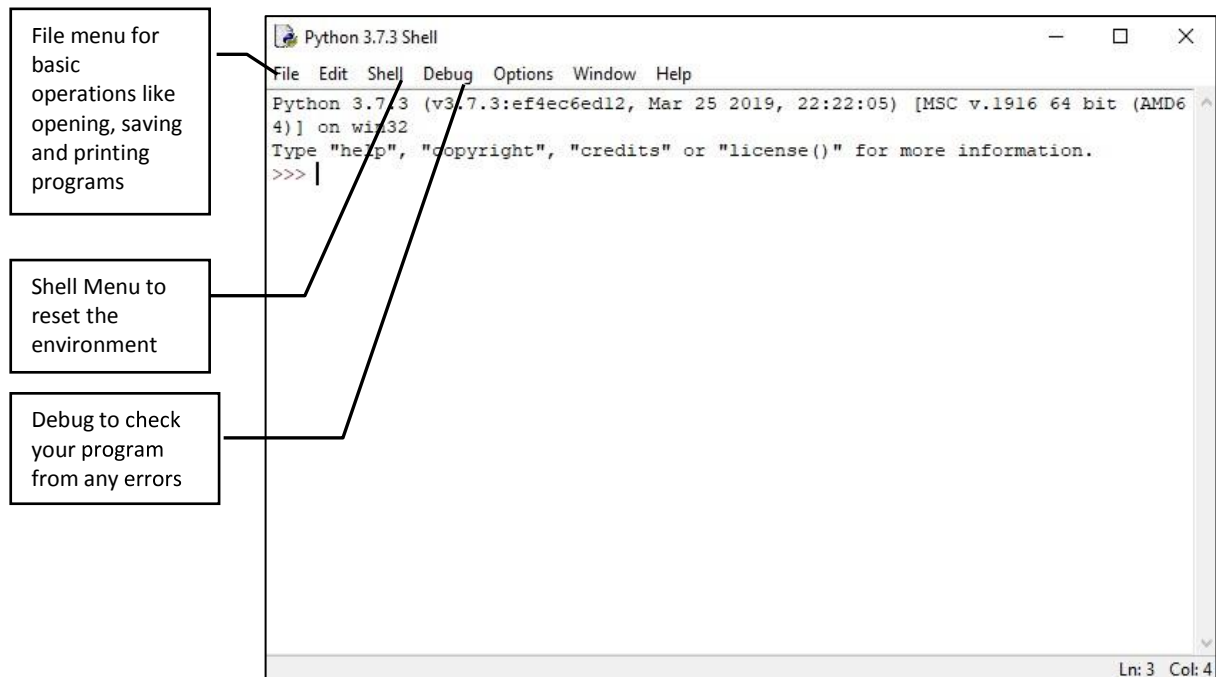
Python is a high-level programming language designed to be easy to read and simple to implement. It is open source, which means it is free to use, even for commercial applications. Python can run on Mac, Windows, and Unix systems.

Python is considered a scripting language, like Ruby or Perl and is often used for creating Web applications and dynamic Web content.

There are tens of thousands of famous python websites on the internet, few of them are listed:

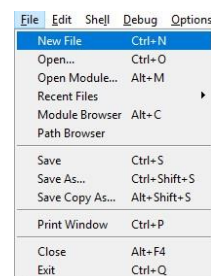
- **Google**
- **Uber**
- **Reddit**
- **Netflix**
- **Dropbox**
- **Facebook**
- **Spotify**
- **Pinterest**
- **Amazon**

Python is a Programming language is popular because of its easy syntax. We will start writing some code in PYTHON IDLE (Integrated Development and Learning Environment).

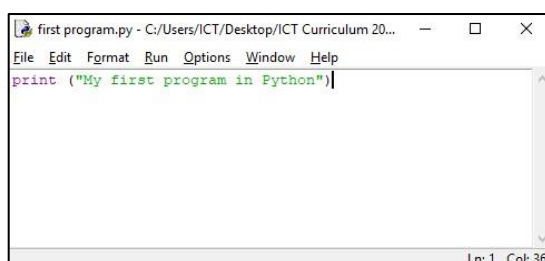


Writing a program in python:

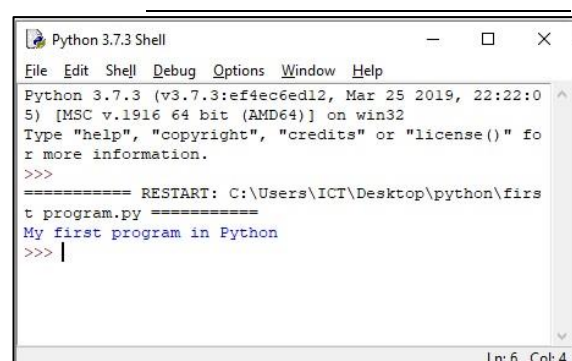
- Open **File** menu and select new file or press Ctrl+N.
- Type → print ("My first program in Python")
- Open **File** menu and save the program or press Ctrl+S.
- Run the program from the **Run** menu of press Ctrl+F5.



Program written python



Output of Program



Reinforcement Handout: Programming Computer - Python

Python Indentations:

Where in other programming languages the indentation in code is for readability only, in Python the indentation is very important. Python uses indentation to indicate a block of code.

```
if 5 > 2:
    print("Five is greater than two!")
```

Python will give an error if we skip the indentation:

```
if 5 > 2:
print("Five is greater than two!")
```

What Are Variables And Their Data Types:

Variables is a block of memory where we store data. Data can be of any type like text, numbers, integers, etc. a variable can store values only in one data type either it could be numeric or string. Python has no command for declaring variable.

Numeric Data Type: This data type is used to hold numeric values like, integers or Float like, decimal numbers.

String Data Type: The string is a sequence of characters like a simple text "Hello World". Python supports Unicode characters. Generally, strings are represented by either single or double quotes.

Python Variables: In Python variables are created the moment we assign a value to it:

```
x = 5
y = "Hello, World!"
```

Comments:

Python has commenting capability for the purpose of in-code documentation. In other words comments are just a dead piece of code which can be used for our references only. Comments start with a #, and Python will render the rest of the line as a comment:

```
#This is a comment.
print("Hello, World!")
```

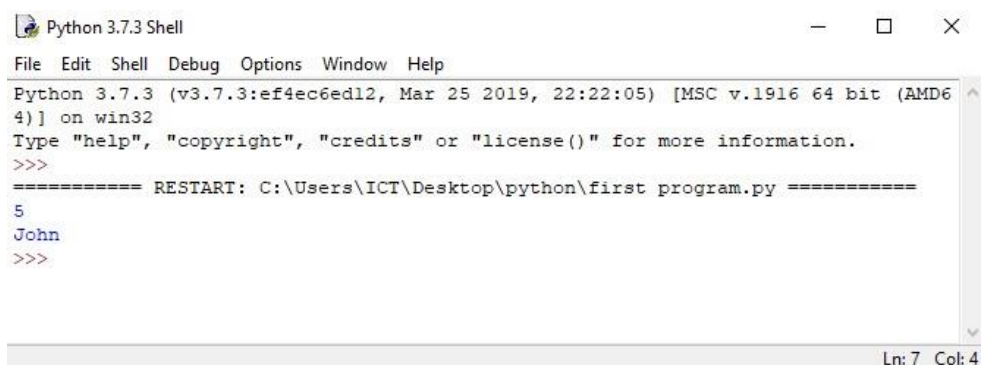
Creating Variables:

Variables are containers for storing data values. Unlike other programming languages, Python has no command for declaring a variable. A variable is created the moment we first assign a value to it.

Declaring variables

```
x = 5
y = "John"
print(x)
print(y)
```

Output Programs



```
Python 3.7.3 Shell
File Edit Shell Debug Options Window Help
Python 3.7.3 (v3.7.3:ef4ec6ed12, Mar 25 2019, 22:22:05) [MSC v.1916 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
===== RESTART: C:\Users\ICT\Desktop\python\first program.py =====
5
John
>>>
```

Ln: 7 Col: 4

Variable Names

A variable can have a short name (like x and y) or a more descriptive name (age, carname, total volume). Rules for Python variables:

- A variable name must start with a letter or the underscore character
- A variable name cannot start with a number
- A variable name can only contain alpha-numeric characters and underscores(A-z, 0-9, and _)
- Variable names are case-sensitive (age, Age and AGE are three different variables)

Reinforcement Handout: Programming Computer - Python

Practice:

1. Open new python file from file menu and write the code as mentioned in the example.
2. Save the document with the name of variables.
3. Run the program from run menu or press Ctrl+F5.
4. Open the variables.py file and add comment using #
5. Save and the run the program and python shell will ignore the comment line.

```
x = 5
y = "John"
print(x)
print(y)
#some random comment
```

← Example Code

Assign Value to Multiple Variables:

Python allows us to assign values to multiple variables in one line:

```
x, y, z = "Orange", "Banana", "Cherry"
print(x)
print(y)
print(z)
```

```
Python 3.7.3 (v3.7.3:ef4ec4) on win32
Type "help", "copyright", "credits() or "license()" for more
>>>
===== RESTART: C:/Python37/Python37-Shell.exe
>>>
Orange
Banana
Cherry
>>>
```

Multiple values can be assign to same variable in one line:

```
x = y = z = "Orange"
print(x)
print(y)
print(z)
```

```
Python 3.7.3 (v3.7.3:ef4ec4) on win32
Type "help", "copyright", "credits() or "license()" for more
>>>
===== RESTART: C:/Python37/Python37-Shell.exe
>>>
Orange
Orange
Orange
>>>
```

Output Variables

The Python print statement is often used to output variables. To combine both text and a variable, Python uses the + character:

```
x = "awesome"
print("Python is " + x)
```

We can also use the + character to add a variable to another variable:

```
x = "Python is "
y = "awesome"
z = x + y
print(z)
```

```
>>>
===== RESTART: C:/Python37/Python37-Shell.exe
>>>
Python is awesome
>>>
```

For numbers, the + character works as a mathematical operator:

```
x = 5
y = 10
print(x + y)
```

If we try to combine a string and a number, Python will give an error.

Python Numbers:

There are three numeric types in Python:

- int
- float
- complex

```
x = 1 # int
y = 2.8 # float
z = 1j # complex
```

Variables of numeric types are created when we assign a value to them:

To verify the type of any object in Python, use the type () function:

- Int, or integer, is a whole number, positive or negative, without decimals, of unlimited length.
- Float, or "floating point number" is a number, positive or negative containing one or more decimals.

```
x = 1 # int
y = 2.8 # float
z = 1j # complex

print(type(x))
print(type(y))
print(type(z))

>>>
===== RESTART: C:/Python37/Python37-Shell.exe
>>>
<class 'int'>
<class 'float'>
<class 'complex'>
>>>
```

Reinforcement Handout: Programming Computer - Python

Type Conversion

We can convert from one type to another with the int() and float() methods:

Example Code

```
x = 1 # int
y = 2.8 # float

#convert from int to float:
a = float(x)

#convert from float to int:
b = int(y)

print(a)
print(b)

print(type(a))
print(type(b))
```

Output

```
1.0
2
<class 'float'>
<class 'int'>
>>>
```

Calculations with numbers

We can also use Python to do any kind of calculation: addition, subtraction, multiplication, division, etc. The rules that we have learned in Microsoft Excel for the use of parentheses apply here, too. For examples, let's suppose that we want to display the term percentage of a student for a subject using average in exam marks and course work.

Multiplications and divisions are calculated before additions and subtractions. This means that $4+2*5$ gives you 14 and not 30. Use parentheses to prioritize the sequence of calculations.

For this example we will also look into the user interaction using **INPUT** feature of python to take data from user end.

```
# Calculate the percentage of a subject
Subject=(input("Enter the subject name:"))
Exam=float(input("Enter the marks obtain in English Exam:"))
CW=float(input("Enter the marks obtain in course work:"))
Percentage= (Exam+CW)/2
print("The Averaage Percentage of", Subject,":", Percentage)
```

In this example first line is having a comment

2. In 2nd line we have declared a variable named Subject with user INPUT to take subject name form user.
3. In 3rd line we have declared a variable name Exam while setting its data type to float.
4. In 3rd line we have declared a variable for course work with the name of CW while setting its data type to float.
5. In 5th line we declare the variable name Percentage while setting its value to a mathematical calculation of Exam+CW divided by 2 to get the average percentage of the subject.
6. In 6th line we have printed the string value along with the values of Subject variable and percentage variable.

Perform the same exercise step by step for practice.

LIST in Python:

In Python you can store your data into variables, but you can also put them in lists. A list is just an ordered collection of items which can be of any data type. Creating a list is as simple as putting different comma separated values between square brackets. Each element of a list is assigned a value by using an index.

An example of a list could be:

```
NewList=[10,20,30, "Samsung"]
```

To call a list element is very easy as calling a cell reference in excel:

```
NewList=[10,20,30, "Samsung"]
print(NewList[3])
```

By this code python output will be Samsung, as the count in list is started from 0.

Delete and add list elements:

Del command is used to delete a list element as mentioned in example below:

Code

```
NewList=[10,20,30, "Samsung"]
print(NewList[3])
del NewList[1]
print(NewList)
```

Output

```
==== RESTART: C:/Us
Samsung
[10, 30, 'Samsung']
>>>
```

Reinforcement Handout: Programming Computer - Python

To add an item to the end of the list, use the **append()** method:

Code

```
thislist = ["apple", "banana", "cherry"]
thislist.append("orange")
print(thislist)
```

Output

```
==== RESTART: C:/Users/ICT/AppData/Local
['apple', 'banana', 'cherry', 'orange']
>>>
```

Python Conditions and If statements

These conditions can be used in several ways, most commonly in "if statements" and loops.

IF

An "if statement" is written by using the **if** keyword.

Code

```
a = 33
b = 200
if b > a:
    print("b is greater than a")
```

Output

```
===== RESTART:
b is greater than a
>>>
```

In this example we use two variables, a and b, which are used as part of the if statement to test whether b is greater than a. As a is 33, and b is 200, we know that 200 is greater than 33, and so we print to screen that "b is greater than a". Indention is necessary, if we do not use the indention as mentioned in example python will give error.

Elif for multiple conditions

The elif keyword is python's way of saying "if the previous conditions were not true, then try this condition".

Code

```
a = 33
b = 33
if b > a:
    print("b is greater than a")
elif a == b:
    print("a and b are equal")
```

Output

```
===== RESTART:
a and b are equal
>>>
```

In this example a is equal to b, so the first condition is not true, but the elif condition is true, so we print to screen that "a and b are equal".

Else

The else keyword catches anything which isn't caught by the preceding conditions.

Code

```
a = 200
b = 33
if b > a:
    print("b is greater than a")
elif a == b:
    print("a and b are equal")
else:
    print("a is greater than b")
```

Output

```
===== RESTART:
a is greater than b
>>>
```

In this example a is greater than b, so the first condition is not true, also the elif condition is not true, so we go to the else condition and print to screen that "a is greater than b". We can also use the else without using elif.

Conditional Operators & Logical Operators:

Conditional Operators

Operator	Meaning
==	Equal to
>	More than
<	Less than
>=	More than or equal to
<=	Less than or equal to
!=	Not Equal to

Logical Operators

Operator	Meaning
and	Both side must be true
or	One Side or other must be true
not	Negates truth

Reinforcement Handout: Programming Computer - Python

Python For Loops

A for loop is used for repeating over a sequence (that is either a list or a string). This is less like the 'FOR' keyword in other programming languages, and works more like an iterator method as found in other object-orientated programming languages.

For example we have a list of students and we want to display the student with the highest marks without using the max() function. We will use the following code:

```
StdMrks=[70, 80, 92.5, 60.2]
MaxMrks=0
for i in range(0,4):
    if StdMrks[i]>MaxMrks:
        MaxMrks=StdMrks[i]
print("Highest Student Marks are:", MaxMrks)
```

1. In 1st line of this code we have created a list **StdMrks** with four values stored in it.
2. In 2nd line we have declared a variable name **MaxMarks** with an integer value of 0.
3. In 3rd line we use for loop while declaring another variable **i** with the range of four. This means this loop will run 4 times. Every time for loop run it self it will increases the value of **i** variable.
4. In 4th line we have set a condition to check that if **StdMrks[i]** variable is greater than **MaxMrks** (declared 0 in 2nd line) variable then change the value of MaxMarks[i] to StdMrks value.
(**StdMrks[i]** variable is going to change its value every time the loop runs and change the index of **StdMrks**. This process is also known as unary increment)
Using indention is compulsory or python will not consider the **for loop** elements and give error.
5. In 5th line **MaxMrks** is setting its value equal to the current value of **StdMarks** only if the condition in previous line is true.
6. In 6th line we have just printed the value of **MaxMrks** along with a string sentence.

Python Functions

In Python, function is a group of related statements that perform a specific task. Functions help break our program into smaller and modular chunks. As our program grows larger and larger, functions make it more organized and manageable.

Furthermore, it avoids repetition and makes code reusable. Function name cannot have spaces in between. Instead of spaces use **_** underscore to connect the words.

In Python a function is defined using the **def** keyword and for executing the function we can use the function name along with parenthesis ().

Code	Output
<pre>def my_function(): print("Hello from a function") my_function()</pre>	<pre>===== RESTART: 0 Hello from a function >>></pre>

Above shown is a function which consists of following components.

- Keyword **def** marks the start of function header.
- A function name to uniquely identify it. Function naming follows the same rules of writing identifiers in Python.
- Parameters (arguments) through which we pass values to a function. They are optional.
- A colon (:) to mark the end of function header.
- Last line executes the function, we can call or use the function in our code wherever it is needed.

Example & Practice:

```
print("Enter your Name: ")
YourName=input ()
print("Hi "+YourName+"!")
```

In this example the input function takes no parameters. So this operation informs the computer to wait until you type your name and press Enter. When you press Enter, the program reads what you have typed and with the print function, it displays a new text string.

Example & Practice:

```
password=input("Type your password: ")
number=len(password)
if number>6:
    print("Your password is strong!")
else:
    print("Your password is weak!")
```

On the other hand, if our function uses an external value, we have to put the values of the parameters in parentheses. For example, type this code and enter Pakistan1248 as a password:

Converting code into a function:

We have already experience this code earlier but now we have converted the same into a function and now it can be recalled and reused whenever required in the program.

```
def marksheet():
    Subject=(input("Enter the subject name:"))
    Exam=float(input("Enter the marks obtain in English Exam:"))
    CW=float(input("Enter the marks obtain in course work:"))
    Percentage= (Exam+CW)/2
    print("The AVERAGE Percentage of", Subject,":", Percentage)

marksheet()
```

Create shapes and graphics with Python (Tkinter):

The Canvas widget supplies graphics facilities for Tkinter. Among these graphical objects are lines, circles, images, and even other widgets. With this widget it's possible to draw graphs and plots, create graphics editors, and implement various kinds of custom widgets and for using those we need to call/import the library of Tkinter so we can use these functions.

We will draw our first example, drawing a line.

The method `create_line(coords, options)` is used to draw a straight line. The coordinates "coords" are given as four integer numbers: `x1, y1, x2, y2` this means that the line goes from the point `(x1, y1)` to the point `(x2, y2)`.

Code

```
from tkinter import *
master = Tk()

canvas_width = 80
canvas_height = 40
w = Canvas(master,
            width=canvas_width,
            height=canvas_height)
w.pack()

y = int(canvas_height / 2)
w.create_line(0, y, canvas_width, y, fill="#476042")

mainloop()
```

Output

For creating rectangles, we have the method `create_rectangle(coords, options)`. Coords is again defined by two points, but this time the first one is the top left point and the bottom right point of the rectangle.

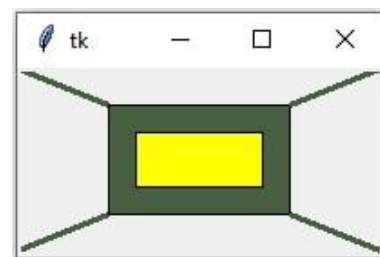
Code

```
from tkinter import *
master = Tk()

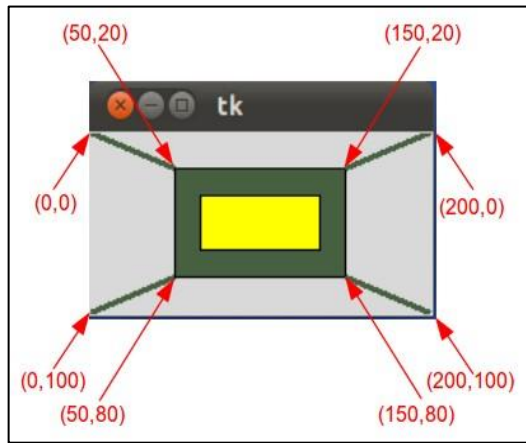
w = Canvas(master, width=200, height=100)
w.pack()

w.create_rectangle(50, 20, 150, 80, fill="#476042")
w.create_rectangle(65, 35, 135, 65, fill="yellow")
w.create_line(0, 0, 50, 20, fill="#476042", width=3)
w.create_line(0, 100, 50, 80, fill="#476042", width=3)
w.create_line(150, 20, 200, 0, fill="#476042", width=3)
w.create_line(150, 80, 200, 100, fill="#476042", width=3)

mainloop()
```

Output

The following image with the coordinates will simplify the understanding of application of create_lines and create_rectangle in our previous example.



We can create an oval on a canvas c with the following method:

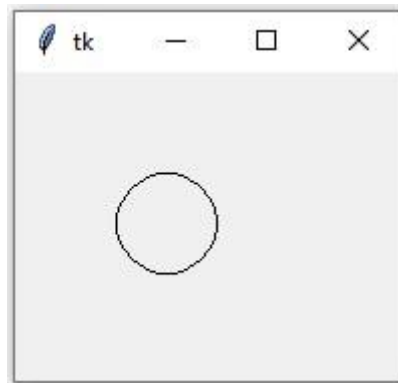
id = C.create_oval (x0, y0, x1, y1, option, ...)

This method returns the object ID of the new oval object on the canvas C. The following script draws a circle around the point (50,50) with the radius 100:

Code

```
from tkinter import *
canvas_width = 190
canvas_height = 150
master = Tk()
w = Canvas(master,
            width=canvas_width,
            height=canvas_height)
w.pack()
w.create_oval(50,50,100,100)
mainloop()
```

Output



Events:

Events in python are more likely we have in other programming languages. They also executes at any specific action/signal occurred for i.e. hovering a mouse at certain point, clicks of mouse either right click or left click and so on.

A Tkinter application runs most of its time inside an event loop, which is entered via the mainloop method. It waiting for events to happen. Events can be key presses or mouse operations by the user. Tkinter provides a mechanism to let the programmer deal with events. For each widget, it's possible to bind Python functions and methods to an event.

widget.bind(event, handler)

If the defined event occurs in the widget, the "handler" function is called with an event object, describing the event.

Code

```
from tkinter import *
def hello(event):
    print("Single Click, Button-1")
def quit(event):
    print("Double Click, so let's stop")
    import sys; sys.exit()

widget = Button(None, text='Mouse Clicks')
widget.pack()
widget.bind('<Button-1>', hello)
widget.bind('<Double-1>', quit)
widget.mainloop()
```

Output

```
===== RESTART: C:/Users/...
Single Click, Button-1
Single Click, Button-1
Single Click, Button-1
Single Click, Button-1
Double Click, so let's stop
>>> Double Click, so let's stop
Single Click, Button-1
Double Click, so let's stop
Single Click, Button-1
Single Click, Button-1
```



In this program event will be trigger by click on the Tkinter widget window. If we click once it will say "Single Click, Button-1" as defined in string values and if we double click it will say "Double click, so let's stop" as defined in string value.

Let's have another simple example, which shows how to use the motion event, i.e. if the mouse is moved inside of a widget:

```

from tkinter import *

def motion(event):
    print("Mouse position: (%s %s)" % (event.x, event.y))
    return

master = Tk()
whatever_you_do = "Expect the best, Prepare for the worst.\n(Muhammad Ali Jinnah)"
msg = Message(master, text = whatever_you_do)
msg.config(bg='lightgreen', font=('times', 24, 'italic'))
msg.bind('<Motion>',motion)
msg.pack()
mainloop()

```

Every time we move the mouse in the Message widget, the position of the mouse pointer will be printed. When we leave this widget, the function motion() is not called anymore.

Basic color list plot for python. There are more colors which can be used with basic words like forestgreen or lime etc.



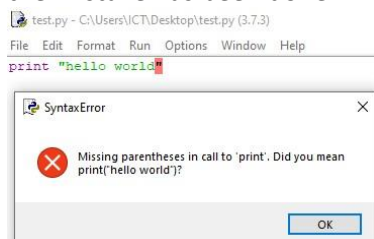
Checking for Bugs:

A software bug is a coding error that causes an unexpected defect in a computer program. In other words, if a program does not perform as intended, it is most likely because of a bug.

There are bugs in software due to unclear or constantly changing requirements, software complexity, programming errors, timelines, errors in bug tracking, communication gap, documentation errors, deviation from standards etc.

There are two types of errors/bug in python or any other programming language.

Syntax Error: Errors in typing the commands and variables. Syntax errors will be automatically detect by the Python IDLE and will show you the error in a dialogue box with suggested solution or the line number where the mistake has been done:



Logical Error: A logical mistake while designing the program which occurs due to the improper planning of the program flow. Logical errors can be avoid working on a Data Flow Diagram (DFD), one practice of DFD is also a flowchart which is an extract of algorithm. Mentioned below are the few areas which can be used to avoid logical errors in your program:

- Form a hypothesis or two before looking at code.
- Resolve syntax errors.
- Start the debugger.
- Identify key variables or conditions.
- Step to your suspicious code.
- Look at the relevant variables.
- Predict what the suspicious line should do.
- Compare your expectations with reality.
- Think about your logic.