2/20/2023

Mathematics Association of Nairobi University
isaak@students.uonbi.ac.ke

In [52]:
```
# Create a github account before today's class
```

Functions

A function is a block of code which only runs when it is called. You can pass data, known as parameters, into a function. A function can return data as a result.

Creating a Function

In Python a function is defined using the def keyword:

In [10]:
```
#Example
def my_function():
 print("Hello from a function")


my_function()
```

```
Hello from a function
```

Arguments

Information can be passed into functions as arguments.

Arguments are specified after the function name, inside the parentheses. You can add as many arguments as you want, just separate them with a comma.

The following example has a function with one argument (fname). When the function is called, we pass along a first name, which is used inside the function to print the full name:

In [11]:
```
def my_function(fname):
 print(fname + " is a Student at the university of Nairobi")
my_function("Emil")
my_function("Tobias")
my_function("Linus")
```

```
Emil is a Student at the university of Nairobi
Tobias is a Student at the university of Nairobi
Linus is a Student at the university of Nairobi
```

In [12]:
```
my_function("Albert")
```

```
Albert is a Student at the university of Nairobi
```

In [14]:
```
# Let's define a mathematic formula using a function, We keep multiplying the base num with pow_num (for loop)
def raise_to_power(base_num, pow_num):
    results = 1
    for index in range(pow_num):
        results *= base_num
    return results
```

In [15]:
```
raise_to_power(10, 2)
```

Out[15]:
```
100
```

In [16]:
```
#function for calculating Cylinder volume
```

```
def cylinder_volume(height, radius):
    pi = 3.14159
    return height * pi * radius ** 2
```

In [17]:
```
#After defining the cylinder_volume function, we can call the function like this.

cylinder_volume(10, 3)
```

Out[17]:
```
282.7431
```

In [21]:
```
height = int(input("Enter the height of the cylinder: "))
radius = int(input("Enter the radius of the cylinder: "))

print("The volume of your cylinder is: ", cylinder_volume(height, radius))
```

```
Enter the height of the cylinder: 12
Enter the radius of the cylinder: 2
The volume of your cylinder is:  150.79631999999998
```

Python Classes and Objects

A `Class` is like an object constructor, or a "blueprint" for creating objects.

Create a Clas

s To create a class, use the keyword `class`:

Example:

In [23]:
```
#Create a class named MyClass, with a property named x:

class MyClass:
  x = 5
```

Create Object

Now we can use the class named MyClass to create objects:

Example Create an object named p1, and print the value of x:

In [24]:
```
p1 = MyClass()
print(p1.x)
```

```
5
```

The `__init__()` Function

The examples above are classes and objects in their simplest form, and are not really useful in real life applications.

To understand the meaning of classes we have to understand the built-in `__init__()` function.

All classes have a function called `__init__()`, which is always executed when the class is being initiated.

Use the `__init__()` function to assign values to object properties, or other operations that are necessary to do when the object is being created:

Example

Create a class named Person, use the `__init__()` function to assign values for `name` and `age`:

In [33]:
```
class Person:
  def __init__(self, name, age):
    self.name = name
    self.age = age

p1 = Person("John", 36)
p2 = Person("Jane", 42)
p3 = Person("Peter", 22)

print(p1.name)
print(p3.age)
```

```
John
22
```

In [41]:
```
class studentz:
    def __init__(self, gender, reg_no, year_of_study, faculty, gpa):
        self.gender = gender
        self.reg_no = reg_no
```

```
        self.year_of_study = year_of_study
        self.faculty = faculty
        self.gpa = gpa

john_Doe = student("Male", "X32/345235/2022", 3, "Main Campus", 2.2)
jane_Doe = student("Female", "F16/234234/2022", 5, "Kikuyu Campus", 3.5)

print(john_Doe.gpa)
print(jane_Doe. faculty)
```

```
2.2
Kikuyu Campus
```

In [ ]:

```
# Quick project
# Building a Multiple-Choice quiz for students
class QUESTIONS:
    def __init__(self, prompt, answer):
        self.prompt = prompt
        self.answer = answer



username = input("Enter your name: ")
reg_no = input("Enter your Registration Number: ")
print("\nThanks " + username + ", Below is your quick test\nSelect the synonym of each of the following words:

question_prompts = [
    "Vicissitude\n (a) sorrows\n(b) misfortunes\n(c) changes\n(d) surprises\n\n",
    "Epitome\n (a) Precise\n(b) Summary\n(c) Spurn\n(d) Exemplar\n\n",
    "Imbecile\n(a) Sane\n(b) Astute\n(c) Foolish\n(d) Aid\n\n",
    "Abeyance\n(a) Suspension\n(b) Persistence\n(c) Continuation\n(d) Rigid\n\n",
    "Yokel\n(a) Intrigue\n(b)Simple-minded\n(c)Victorious\n(d)Noise\n\n",

]
QUIZS = [
    QUESTIONS(question_prompts[0], "c"),
    QUESTIONS(question_prompts[1], "d"),
    QUESTIONS(question_prompts[2], "c"),
    QUESTIONS(question_prompts[3], "a"),
    QUESTIONS(question_prompts[4], "b"),
]


def run_test(QUESTIONS):
    score = 0
    for question in QUIZS:
        answer = input(question.prompt)
        if answer == question.answer:
            score += 1
    print("At " + username + " You got " + str(score) + "/" + str(len(QUIZS)) + " correct in your test.")


run_test(QUESTIONS);

'''
question_prompts = [
    "Vicissitude\n (a) sorrows\n(b) misfortunes\n(c) changes\n(d) surprises\n\n"
    "Epitome\n (a) Precise\n(b) Summary\n(c) Spurn\n(d) Exemplar\n\n"
    "Imbecile\n(a) Sane\n(b) Astute\n(c) Foolish\n(d) Aid"
    "Abeyance\n(a) Suspension\n(b) Persistence\n(c) Continuation\n(d) Rigid\n\n"
    "Yokel\n(a) Intrigue\n(b)Simple-minded\n(c)Victorious\n(d)Noise\n\n"
]

'''
```

python modules

A `python module` can be defined as a python program file which contains a python code including python functions, class, or variables.

In other words, we can say that our python code file saved with the extension (`.py`) is treated as the module. We may have a runnable code inside the python module.

Modules in Python provides us the flexibility to organize the code in a logical way.

Loading the module in our python code:

We use:

1. The `import` statement
2. The `from-import` statement

The import statement

The `import` statement is used to import all the functionality of one module into another. Here, we must notice that we can use the functionality of any python source file by importing that file as the module into another python source file.

We can `import` multiple modules with a single import statement, but a module is loaded once regardless of the number of times, it has been imported into our file.

The from-import statement

Instead of importing the whole module into the namespace,python provides the flexibility to import only the specific attributes of a module. This can be done by using `from < module-name> import <name 1>, <name 2>..,<name n>` statement.

In [ ]: