PYTHON CHEAT SHEET

1. print(“ “) - to print something
2. age=10 - {use of variable}
3. age=input(“what is your name?”) - {to input a string}
4. ALWAYS KEEP SPACE BETWEEN OPERANDS AND OPERATORS
5. age=int(age) - conversion of input taken
6. print(type(age))-u get the datatype
7. When you take input u always get in string
8. Double quotes and single quotes can be interchanged in strings as an escape sequence
   For e.g “python’s beginner course”
   ‘He said “i love football”
9. Use ‘’’
   Alexio
   Aiden
   Alan
   ‘’’
   For multiline text
10. Index
    With e.g.
    class=”First Class”
    print(class[1])
    F
    print(class[-1])
    S-{character from the last}

11.substring in python
print(class[0:3])
NOTE:similar to java only prints till the index before the given index and starts with the given index
Output:
Fir

Exception-
print(class[2:])/(upto the end)/
rst Class
print(class[:5])/(from the start)/
First
print(class[1:-1])(including the 1st index {excluding the 0 index}excluding the -1 index)

irst clas

12.inserting or concatenating two strings{note f is really important}

```
name='aexio'
name_2='jose'
full_name=f'{name} {name_2}'
print(full_name)
```

aexio jose

13.print(len(class))
  Prints the length
  11

14.String methods

- class.upper() makes the whole string uppercase it does not change the value in the variable though
   print(class.upper())
   FIRST CLASS
   print(class)
   First Class

- class.lower()is similar

- class.find('i')returns the index of the character in the string{it is sensitive to case}
  It returns -1 if character not present
  It can also be used for a set of characters like
  class.find('Class')

  Returns the index of C which is 6
  class.replace() replaces set of characters in a string

```
name='alexio jose'
name_2='ajeesh'
full_name=f'{name} {name_2}'
print(full_name.replace('jose','joseee'))
```

  alexio joseee ajeesh

```
print('jose' in name)
```

  True

15. Functions are general while methods are specific to a particular object or data type like string methods not string functions

16.normal operators in python and shorthand is also there (no increment /decrement operator)

Exception

//-used to return only integral part of quotient

**-exponentiation

Order of precedence of operators

**

/ or *

+ Or -

17. Math functions

round

abs

18.to use math functions such as ceil and floor

U have to import math module

```
import math
print(math.ceil(3.5))
```

ALL MATH MODULE FUNCTIONS

https://www.w3schools.com/python/module_math.asp

19. Program using if ,elif,else

```
    a=int(input("enter a number"))
b=int(input("enter another number"))
if(a>b):
 print("the first number is greater")
 print("the greater number is :"+ f'{a}')
elif(a<b):
 print("the second number is greater")
 print("the greater number is :"+ f'{b}')
else:
 print("Both numbers are equal")
```

20. In python logical operators have no symbols but are literally and ,not,or

21. Relational operators are the same

22.while condition:

(indentation) statements

23.while True:

Infinite loop

```
24. for item in 'PYTHON':
        print (item)
```

```
 Output
P
Y
T
H
O
N
```

This works for one string since string is an array of characters

Same applies for a list of names and numbers

```
 for item in[1,2,3,4]:
    print(item)
```

```
Output
1
2
3
4
```

25.python has a built in function called range(excluded number) to print numbers till before a certain number

```
for item in range(10):
     print(item)
```

```
Output:
0
1
2
3
4
5
6
7
8
9
```

```
for item in range(5,10):
	print(item)
```

Output:
```
5
6
7
8
9
```

To skip values
```
for item in range(5,10,2):
	print(item)
```

Output:
```
5
7
9
```
26.to assign an array
 E.g.
 numbers=[1,2,3,4]
27.to print
```
xxxxx
xx
xxxxx
xx
Xx
```

```
numbers=[5,2,5,2,2]
for mas in numbers:
   print("x"*mas)
```

Without *function
27.list methods-{numbers=[1,2,3,4] this is a list}:output
 numbers.append(20)-adds twenty to the end:{1,2,3,4,20}
 numbers.insert(1,25)-adds 25 to the first index:{1,25,2,3,4}
 numbers.remove(3)-removes the number 3, not the index :{1,2,4}
 numbers.clear()-removes all the list entities:{}
 numbers.pop()-removes the last digit:{1,2,3}
 numbers.index(2)-gives the index of the first appearance of 2: 1(if number not in the list error appears)
 50 in numbers-returns a boolean value:False
 2 in numbers-returns a boolean value:True
 numbers.count(5)-returns the number of 5s in the list

numbers.sort()-arranges the values of the list in ascending order and returns None(None shows the absence of any value)
x=numbers.copy()-just copies to another variable
For list numbers[1,2,3,4]
10 not in numbers :True

28.Tuples(like list but cannot be mutated or changed)
Defined as numbers=(1,2,3){use of parentheses)
They only have two methods
numbers.count(1):1
numbers.index(3):2
numbers[0]=100 is not possible and shows errors as tuples cannot be mutated
We can assign tuple values to variables

```
x=numbers[2]
print(x)
```

Output

```
3
```

29.dictionaries
Used for key value pairs

```
customer={
  "name":"John Smith",
  "is_verified":True,
  "age":32,
}
print(customer["name"])
```

Output:

```
John Smith
```

Similarly u can do age and it returns 32 but if u give a keyword that does not exist then it shows an error and the same keyword cannot be used twice
And python is case sensitive

If u use

print(customer.get(“birthdate”))-use parenthesis
Then it does not show an error but returns None
Similar to strings and lists
The values of key words can be updated

```
customer[“name”]=”jack smith”
print(customer[“name”])
```

Output:
jack smith

New key pairs can also be added
customer[“birthdate”]=”26th september,1983”
print(customer[“birthdate”])

Output:
26th september,1983

29.**to print in one line use a variable and keep concatenating to the variable**
output+=blah blah bah
30.message=”alexio jose ajeesh”
message.split(“ “)
So split separates different words using the character given in the parenthesis as boundary and **returns a list**
So u can assign it to a variable
words=message.split(“ ”)
31.
output+=digits.get(numbers, numbers)+" "
When you do this instead of printing None it returns the given value (it is stored as the default value to be printed instead of none)
For e.g.

```
phone=input("phone:")
digits={
 "1": "One",
 "2": "Two",
 "3": "Three",
 "4": "Four",
 "5": "Five",
 "6": "Six",
 "7": "Seven",
 "8": "Eight",
 "9": "Nine"
}
output=""
for numbers in phone:
   output+=digits.get(numbers, numbers)+" "
print(output)
```

Here if we give 98886588872ajeeshjose it returns
Nine Eight Eight Six Five Eight Eight Eight Seven Two a j e e s h j o s e

33.
```
def greet_user(name,name_2):
   print(f"hello {name} {name_2}!")
   print("welcome aboard")

 greet_user(name_2="Maria",name=”Joseph”)
```

Output:
```
hello Joseph Maria!
welcome aboard
```

Indentation and leaving two lines is very imp
The def keyword is used to define the function

34.using a return statement
```
def square(x):
   sq=x*x
   return sq
print(square(3))
```

Output:
```
9
```

We can also store it in a variable
```
y=square(3)
```

```
def square(x):
   sq=x*x
print(square(3))
```

Output:
```
9
None
```

It prints none because none is the default value for return
And since there is no return statement
It returns none by default

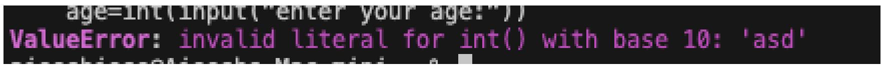
Python can return more than one value

35.

```
try:
   age=int(input("enter your age:"))
   print (age)
except ValueError:
   print("invalid")
```

Here we are trying to make the interpreter to not display an error when the user enters anything other than an integer and instead to print a error message

If we do this without the try and except



```
    age=int(input("enter your age:"))
ValueError: invalid literal for int() with base 10: 'asd'
```

This value error is printed

There can be more than one accept statement for the different types of errors like division by zero

<u>Note:in the except block just take the error that u find at the end of the program and put it in the except condition</u>

ZeroDivisionError: division by zero

So use that error for the exception

```
try:
age=int(input("Age:"))
income=20000
risk=income/age
print(age)
except ValueError:
print("invalid entry")
except ZeroDivisionError:
print("Age cannot be zero")
```

White highlighted part shows how it is used

36.# is used to make a comment line
   Each line should start with a new # for multi line comments

```
#alexio jose ajeesh is the writer
#he is a good buoy
```

37.

- classes are used to define new types of data

- White naming classes the first letter should always be capital and it is known as the pascal naming system.Instead of using underscore to separate different words we use camel casing
  For e.g.,
  class Point:

- We can make multiple functions inside the class for that particular class
- The main purpose of the class is to make objects
- These objects can be added to variables for example

```
class Point:
   def move(self):
       print("move")
   def draw(self):
       print("draw")

point1=Point()
point1.x=10
point1.y=20
print(point1.x)
point1.draw()
point2=Point()
```

- We are creating the object by calling the class like a function

```
Point()
```

- This object is assigned to a variable and multiple objects can be created by using different variable names
- Self must be used inside the function as an argument
- Here x and y are the attributes of the object point1 and if we do this

```
point2=Point()
print(point2.x)
```

  It will show an error as there is no attribute x assigned to the object point 2

<u>38.Classes and Constructors are almost similar to java in function</u>

39.

- Constructors are called during object creation
- self refers to the object itself

```
def __init__(self,x,y):
     self.x=x
     self.y=y
```
And hence when we call the object

We use __(double underscore)init(short for initialization)__to use constructor

```
point1=Point(1,3)
```
We pass values for the compiler to form new attributes
So if we execute this program
```
class Point:
   def __init__(self,x,y):
     self.x=x
     self.y=y
   def move(self):
       print("move")
   def draw(self):
       print("draw")

point1=Point(1,3)
print(point1.x)
point1.draw()
point2=Point(12,13)
print(point2.x)
```

We get
```
1
draw
12
```
As the constructor is called and the value is assigned to the attribute

Note:the initialized variable can be used in other functions as well.

40.INHERITANCE

```
 class Person:
 def speak(self):
   print("I speak")


class Boy(Person):
 pass


class Girl(Person):
 pass

boy1=Boy()
boy1.speak()
girl1=Girl()
girl1.speak()
```

- Here we are reducing the repeating of the speak function for every class by inheritance
- The pass is used to not leave in any function empty
- Instead of using the pass function

```
class Person:
 def speak(self):
   print("I speak")


class Boy(Person):
 def speaktype(self):
   print("with a bass voice")

class Girl(Person):
 def hair(self):
   print("i have long hair")

boy1=Boy()
boy1.speak()
boy1.speaktype()
girl1=Girl()
```

```
girl1.speak()
girl1.hair()
```

```
Output:
I speak
with a bass voice
I speak
i have long hair
```

What happens if you don't use `self`?

If you write a method in a class without `self`, you **can't access instance variables** like `self.name`. The method wouldn't know which data to use.

What is `self` in Python?

In Python, `self` is **just a name**—by convention—that refers to the **current object (instance)** of a class.

You use `self` inside class methods to **access variables and methods** that belong to **that specific object**.

🔧 Why is `self` important?

Because without it, **Python wouldn't know** which object you're referring to when you're inside a method. `self` helps you work with data that belongs to that particular instance of the class.

`self` = the current object.

Lets you access **object-specific** data inside methods.

Required as the **first parameter** in instance methods.

41.Modules are files with python codes which are used to categorize codes
  i)In one file write the following code(for e.g alexiojose)

```
def kgs_lbs(weight):
   return weight*0.45


def lbs_kgs(weight):
   return weight/0.45
```

ii)open a new file and import the previous file

```
import alexiojose
x=alexiojose.kgs_lbs(50)
print(x)
```

Now we can use the functions of the other file
If we run the code it returns
22.5

iii)instead if we use this code

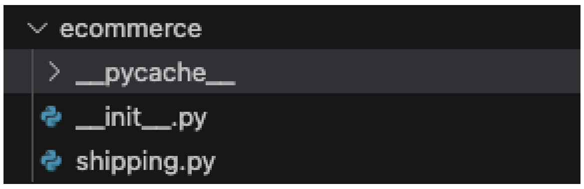
```
from alexiojose import kgs_lbs
x=kgs_lbs(100)
print(x)
```

<u>We do not have to use the module name before the function</u>

The keyword used is from…..import…….

So we can either import the entire module or a particular function form the module.it is done by using import and from…import respectively

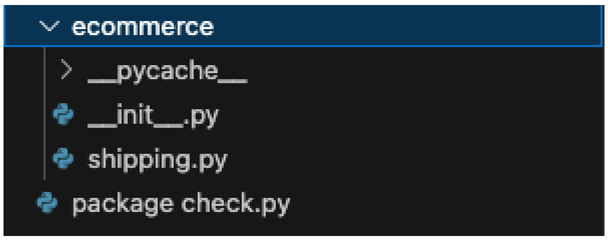
42.packages are a group of categorised modules



1. you form a directory(ecommerce) which you can convert to a python package by adding the module __init__.py
2. then you create a new module(file)for example shipping where u add functions

```
def calculate_shippng():
   print("calc shipping")
```

3.then create a file outside the ecommerce package


```
ecommerce
  __pycache__
  __init__.py
  shipping.py
package check.py
```

Like the package check

4.similar to importing a module you can import either a specific module or a specific function from a module

i)importing a specific module

a)
```
import ecommerce.shipping
ecommerce.shipping.ship_cost()
```

b)
```
from ecommerce import shipping
shipping.ship_cost()
```

ii)importing only a specific function
```
from ecommerce.shipping import ship_cost,ship_speed
ship_cost()
ship_speed()
```
If you want more functions just add a comma

43.generating random values using standard libraries
https://docs.python.org/3/py-modindex.html

To generate random numbers

```
import random
m=int(input("enter a number:"))
n=int(input("enter a number:"))
print(f"three numbers between {m} and {n}")
for x in range(3):
   print(int(random.randint(m,n)))
```

U can use the random.randint()function to generate random numbers in a given range

```
import random
names=["mosh","alexio","john","alan","ashaz"]
leader=random.choice(names)
print(f"your leader is {leader}")
```

This returns any random value from the given list

- PEP-python enhancement proposal

44.

```
from pathlib import Path
path=Path("emails")
print(path.mkdir())
```

The pathlib directory has a module called path which has multiple functions

So here we are using it(path.mkdir) to create a new directory which can be converted to a python package by adding a __init__.py file

```
from pathlib import Path
path=Path("emails")
print(path.rmdir())
```

Here we are using path.rmdir to remove that same directory
Both return none

We can check if something exists by using

```
from pathlib import Path
path=Path("emails")
print(path.exists())
```

It returns a boolean value

```
from pathlib import Path
path=Path("ecommerce")
for file in path.glob("*"):
   print(file)
```

This returns all the files in the directory{the * sign signifies all files}

ecommerce/__init__.py
ecommerce/__pycache__
ecommerce/shipping.py

If you make it path.glob(“*.py”)it returns all the python files

45. There are many directories which have been uploaded on the web called python package index which u can use

✅ What You Need First

1️⃣ Python installed

Run this in the VS Code terminal:

bash
CopyEdit
```
python3 --version
```

If it shows something like `Python 3.11.7`, you're good! 🎉
 If not, [download Python](https://www.python.org) and install it.

📦 How to Use PyPI (Python Packages) in VS Code on Mac

🔹 STEP 1: Open Terminal in VS Code

- Open your Python file or folder in VS Code
- Press `Ctrl + ~` (or go to **View > Terminal**)
- You'll now see the integrated terminal at the bottom

🔹 STEP 2: Install a Package from PyPI

Use **`python3 -m pip`** to install packages:

bash
CopyEdit
```
python3 -m pip install openpyxl
```

Or any other package, like:

bash
CopyEdit
```
python3 -m pip install requests
python3 -m pip install pandas
```

✅ This installs the package into the global Python environment.

🔹 STEP 3: Use the Package in Your Code

Now in your Python file (e.g., `app.py`), you can write:

python
CopyEdit
```
from openpyxl import Workbook

wb = Workbook()
ws = wb.active
ws['A1'] = "PyPI works!"
wb.save("output.xlsx")
```

Then run it in the terminal:

bash
CopyEdit
```
python3 app.py
```

46. There are two types of path
i)absolute path:from the beginning like the root of the hard disk
c:/documents/ajeeshjose/python/ecommerce/shipping
ii)relative path:from a given file
python/ecommerce/shipping