NEEC

BASIC

#this is a comment
Makes a comment.
help(print)
Gives help about a function.

INPUT AND OUTPUT

print("Hello World", x)
Prints string "Hello World" and the value of the variable x.

input()
Reads string from the stdin.

TYPES

Variables do not need to be declared with any particular type and can even change type after they have been set.

INT
is positive or negative integer of unlimited length).
EX: x=1 or x= int(1)

FLOAT
is positive or negative number containing one or more decimals.
EX: y=2.8 or x=float(2.8)

STRING
is a array of bytes representing unicode characters.
EX: w= "Hello World" or w=str("Hello World")

1- len(x) Returns the number of items (length) of an object. Can be used in any object: lists, sets, etc…
EX: len("Hello, World!"), returns 13
2-("Hello").lower() returns the string "Hello" in lower case.
3-("Hello").upper() returns the string "Hello" in upper case.
4-("Hello").replace("H", "J")
Replaces a string with another string.
5-("Hello").center(width, fillchar)
Center the text and fill the blank spaces with the fillchar.
6- "weight: {} Kg".format(57.2)- move the objects in () to place marked by {}.
7-str.join(sequence)- returns a string in which the string elements of sequence have been joined by str separator.

LIST
is a collection which is ordered and changeable. Allows duplicate members.

EX: thislist = ["apple", "banana", "cherry"] or **thislist=list(("apple", "banana", "cherry"))**

1- print(thislist[1]) - Access the list items by referring to the index number: **prints banana.**
2- thislist.append("orange") - add an item to the end of the list.
3- thislist.insert(1, "orange") - add an item at the specified index.
4- thislist.remove("banana") removes the specified item.
5- thislist.pop(1) removes the specified index, (or the last item if index is not specified).
6- del thislist delete the list completely.
7- thislist.clear() empties the list.
8- thislist.count("banana") returns the number of elements with the specified value.
9- thislist.sort Sorts the list.
10- thislist.extend(vegetables) add the elements of vegetables to the thislist list.
11- thislist.index("banana") returns the index of the first element with the specified value.

TUPLE
is a collection which is ordered and unchangeable. Allows duplicate members.
EX: thistuple = ("apple", "banana", "cherry") or **thistuple=tuple(("apple", "banana", "cherry"))**

1- prints(thistuple[1]) Access the tuple items by referring to the index number: **prints banana.**
2- del thistuple delete the tuple completely.
3- thistuple.count("apple") returns the number of elements with the specified value.
4- thistuple.index("apple") returns the index of the first element with the specified value.

SET
is a collection which is unordered and unindexed. No duplicate members.
EX: thisset = {"apple", "banana", "cherry"} or **thisset = set(("apple", "banana", "cherry"))**

1- thisset.add("orange") Adds an element to the set.
2- thisset.clear() empties the set.
3- thisset.difference(y) Returns a set containing the difference between two or more sets.
4- thisset.intersection(y) Returns a set, that is the intersection of two other sets.
5- thisset.issubset(y) returns whether another set contains this set or not.
6- thisset.issuperset(y) returns whether this set contains another set or not.
7- thisset.pop() removes a random element from the set.
8-thisset.remove("apple") removes the specified element.
9- thisset.union(y) return a set containing the union of sets.

DICTIONARY
is a collection which is unordered, changeable and indexed. No duplicate members.
EX: thisdict = {"Brand": "NEEC", "age": 15, "year" : 2018} or **thisdict = dict(brand="NEEC", age=15, year=2018)**

1- **1-thisdict["age"]** or **thisdict.get("age")** get the value of "model" key.
2- thisdict["year"] = 2018 change the value of a specific item.
3- Thisdict.pop("year") removes the item with the specified key name.
4- del thisdict delete the dictionary completely.
5- del thisdict["year"] removes the item with the specified key name.
6- thisdict.clear()empties the dictionary.

python™

OPERATORS

ARITHMETIC

x + y Addition
x - y Subtraction
x * y Multiplication
x / y Division
x % y Modulus-rest of the division
x ** y Exponentiation- x^y

COMPARISON

x == y Equal
x != y Not Equal
x > y Greater than
x < y Less than
x >= y Greater than or equal to
x <= y Less than or equal to

LOGICAL

x and y Returns True if both statements are true
x or y Returns True if one of the statements is true
not x Reverse the result, returns False if the result is true

IDENTITY

x is y Returns True if both variables are the same object
x is not y Returns True if both variables are not the same object

MEMBERSHIP

x in y Returns True if a sequence with the specified value is present in the object
x not in y Returns True if a sequence with the specified value is not present in the object

CONDITIONS

```
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")
```

Indentation is important here.

LOOPS

For Loops

for x in WS: iterate through the array WS

```
        print(x)
```

print NEEC on the screen.

for x in range(2,6):

```
        if x == 4,
```

condition to continue to the next loop

```
                continue
        print(x)
```

print the numbers 2, 3 and 5 on the screen, doesn't print the number 4.
for item in [2,4,8,10]: iterates through the list [2,4,8,10].
for item in (2,4,8,10): iterates through the tuple (2,4,8,10).
for x in range(2,4): will iterate through the numbers

```
        print (x)
```

print the numbers 2 and 3 on the screen.
for x in range(2,10,3): will iterate through the numbers, 3 by 3

```
        print (x)
```

print the numbers 2, 5 and 8 on the screen.

```
for x in range(2,6)
        if x == 4
                break will stop the
loop
        print(x)
```

print the numbers 2 and 3 on the screen.

While Loops

```
i = 1
while i < 6:
  print(i)
  if i == 3:
    break
  i += 1
```

print the numbers 1 and 2 on the screen.

```
i = 0
while i < 4:
  i += 1
  if i == 2:
    continue
  print(i)
```

print the numbers 1, 3 and 4 on the screen.

FUNCTIONS

```
def first_function()
        print("NEEC")
```

first_function is defined using the def keyword.

```
first_function()
```

whenever the function is call, will print NEEC on the screen.

Lambda

Lambda functions can take any number of arguments.

```
x = lambda a, b : a + b
print(x(5, 6))
```

print the number 11 on the screen.
A lambda function that take 2 arguments and add argument a with argument b and print the result.

CLASSES AND OBJECTS

```
class MyWS:
        x = 20
```

This is how to create a class, with one property (x).

```
pobj = MyWS()
print(pobj.x)
```

This is how to create an object and print the value of x.

The __init__() Function

```
class brand:
 def __init__(brand, name, age):
   brand.name = name
   brand.age = age

p1 = brand("NEEC", 15)

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

print NEEC and the number 15 on the screen.
This is how to create a class named brand, use the __init__() function to assign values for name and age

```
p1.name = NEECIST
```

This is how you can modify properties on objects.

```
del p1.age
```

This is how to delete a property on a Object.

MODULES

This code is saved in the file WSModule.py

```
brand1 = {
 "name": "NEEC",
 "age": 15,
 "country": "Portugal"
}
```

You can import a module like this:

```
import WSmodule
a = WSmodule.brand1["name"]
print(a)
```

print the NEEC on the screen.
Create an alias for mymodule called WS:

```
import mymodule as WS
```

python™