```
#PROBLEM SET 2
#LOOPS
```

camelCase PROBLEM

In some languages, it's common to use camel case (otherwise known as "mixed case") for variables' names when those names comprise multiple words, whereby the first letter of the first word is lowercase but the first letter of each subsequent word is uppercase. For instance, whereas a variable for a user's name might be called name, a variable for a user's first name might be called firstName, and a variable for a user's preferred first name (e.g., nickname) might be called preferredFirstName.

Python, by contrast, recommends snake case, whereby words are instead separated by underscores (_), with all letters in lowercase. For instance, those same variables would be called name, first_name, and preferred_first_name, respectively, in Python.

In a file called camel.py, implement a program that prompts the user for the name of a variable in camel case and outputs the corresponding name in snake case. Assume that the user's input will indeed be in camel case.

Hints

Recall that a str comes with quite a few methods, per docs.python.org/3/library/stdtypes.html#string-methods.

Much like a list, a str is "iterable," which means you can iterate over each of its characters in a loop. For instance, if s is a str, you could print each of its characters, one at a time, with code like:

for c in s:

print(c, end = "")

```
def main():
  camel_case = input("Enter a variable name in camel case: ")
  snake_case = convert_to_snake_case(camel_case)
  print("Snake case:", snake_case)


def convert_to_snake_case(camel_case):
    snake_case = ''
    for char in camel_case:
        if char.isupper():
            snake_case += '_' + char.lower()
        else:
            snake_case += char

    if snake_case.startswith('_'):
        snake_case = snake_case[1:]
    return snake_case



main()
```

```
Enter a variable name in camel case: fuckNmimsShirpur
Snake case: fuck_nmims_shirpur
```

COKE MACHINE PROBLEM

Suppose that a machine sells bottles of Coca-Cola (Coke) for 50 cents and only accepts coins in these denominations: 25 cents, 10 cents, and 5 cents.

In a file called coke.py, implement a program that prompts the user to insert a coin, one at a time, each time informing the user of the amount due. Once the user has inputted at least 50 cents, output how many cents in change the user is owed. Assume that the user will only input integers, and ignore any integer that isn't an accepted denomination.

```
print("Amount Due: 50")
amount_due = 50
coins_added = 0

while True:
    insert_coin = int(input("Insert Coin: "))
    if insert_coin == 25 or insert_coin == 10 or insert_coin == 5:
        amount_due -= insert_coin
        coins_added += insert_coin
    else:
        print(f"Amount Due: {amount_due}")

    if coins_added >= 50:
        print(f"Change Owed: {coins_added - 50}")
        break
    else:
        print(f"Amount Due: {amount_due}")
```

```
Amount Due: 50
Insert Coin: 25
Amount Due: 25
Insert Coin: 10
Amount Due: 15
Insert Coin: 25
Change Owed: 10
```

JUST SETTING UP MY TWTTR PROBLEM

When texting or tweeting, it's not uncommon to shorten words to save time or space, as by omitting vowels, much like Twitter was originally called twttr. In a file called twttr.py, implement a program that prompts the user for a str of text and then outputs that same text but with all vowels (A, E, I, O, and U) omitted, whether inputted in uppercase or lowercase.

Hints

Recall that a str comes with quite a few methods, per docs.python.org/3/library/stdtypes.html#string-methods.

Much like a list, a str is "iterable," which means you can iterate over each of its characters in a loop. For instance, if s is a str, you could print each of its characters, one at a time, with code like:

for c in s: print(c, end="")

```
answer = input("Input: ")
print("Output: ", end = "")

for letters in answer:
    if letters.lower() not in ["a","e","i","o","u"]:
        print(letters, end = "")

print()
```

```
Input: twitter
Output: twttr
```

VANITY PLATES PROBLEM

In Massachusetts, home to Harvard University, it's possible to request a vanity license plate for your car, with your choice of letters and numbers instead of random ones. Among the requirements, though, are:

"All vanity plates must start with at least two letters."

"… vanity plates may contain a maximum of 6 characters (letters or numbers) and a minimum of 2 characters."

"Numbers cannot be used in the middle of a plate; they must come at the end. For example, AAA222 would be an acceptable … vanity plate; AAA22A would not be acceptable. The first number used cannot be a '0'."

"No periods, spaces, or punctuation marks are allowed."

In plates.py, implement a program that prompts the user for a vanity plate and then output Valid if meets all of the requirements or Invalid if it does not. Assume that any letters in the user's input will be uppercase. Structure your program per the below, wherein is_valid returns True if s meets all requirements and False if it does not. Assume that s will be a str. You're welcome to implement additional functions for is_valid to call (e.g., one function per requirement).

def main():

```
plate = input("Plate: ")
if is_valid(plate):
    print("Valid")
else:
    print("Invalid")
```

def is_valid(s): ...

main()

Hints

Recall that a str comes with quite a few methods, per docs.python.org/3/library/stdtypes.html#string-methods.

Much like a list, a str is a “sequence” (of characters), which means it can be “sliced” into shorter strings with syntax like s[i:j]. For instance, if s is "CS50", then s[0:2] would be "CS".

```
def main():
    plate = input("Plate: ")
    if is_valid(plate):
        print("Valid")
    else:
        print("Invalid")


def is_valid(s):
    if len(s) >= 2 and len(s) <= 6:
        if s.isalpha():
            return True
        elif s.isalnum() and s[0:2].isalpha():
            for char in s:
                if char.isdigit():
                    position = s.index(char)

                    if s[position:].isdigit() and int(char) != 0:
                        return True

                    else:
                        return False

main()
```

```
Plate: CS50P
Invalid
```

NUTRITION FACTS PROBLEM

The U.S. Food & Drug Adminstration (FDA) offers downloadable/printable posters that “show nutrition information for the 20 most frequently consumed raw fruits … in the United States. Retail stores are welcome to download the posters, print, display and/or distribute them to consumers in close proximity to the relevant foods in the stores.”

In a file called nutrition.py, implement a program that prompts consumers users to input a fruit (case-insensitively) and then outputs the number of calories in one portion of that fruit, per the FDA’s poster for fruits, which is also available as text. Capitalization aside, assume that users will input fruits exactly as written in the poster (e.g., strawberries, not strawberry). Ignore any input that isn’t a fruit.

Hints

Rather than use a conditional with 20 Boolean expressions, one for each fruit, better to use a dict to associate a fruit with its calories!

If k is a str and d is a dict, you can check whether k is a key in d with code like:

if k in d:

```
  ...
```

Take care to output the fruit's calories, not calories from fat!

```
fruits = {
    "apple": 130,
    "avocado": 50,
    "banana": 110,
    "cantaloupe": 50,
    "grapefruit": 60,
    "grapes": 90,
    "honeydew melon": 50,
    "kiwifruit": 90,
    "lemon": 15,
    "lime": 20,
    "nectarine": 60,
    "orange": 80,
    "peach": 60,
    "pear": 100,
    "pineapple": 50,
    "plums": 70,
    "strawberries": 50,
    "sweet cherries": 100,
    "[illegible]": 50,
```