Basic Programming Review

Checklist

- variables
- control flow
- loops

Environment

- Thonny

Types

- numbers
- strings
- lists
- dictionaries

Numbers

```
2 + 2
50 - 5*6
(50 - 5*6) / 4
8 / 5
17 // 3
17 % 3
5 ** 2
```

Strings

```
'spam eggs"
'1975'
'doesn\'t' or "doesn't"
'"Yes," they said.'
'   "   Isn  \'   t,  "   they said.    '
```

Strings

```
prefix + "thon"
word = 'Python'
word[0]
word[5]
word[-1]
word[0:3]
word[4:0:-2]
```

Lists

```
squares = [1, 4, 9, 16, 25]
cubes = [1, 8, 27, 65, 125]

cubes[3] = 64
cubes.append(216)
cubes.append(7 ** 3)
```

Lists

```
letters = ['a', 'b', 'c', 'd', 'e', 'f', 'g']
letters[2:5]
letters[::2]

letters[2:5] = []
```

Dictionaries

```
tel = {'jack': 4098, 'sape': 4139}
tel['guido'] = 4127
tel['jack']
del tel['sape']
tel['irv'] = 4127
list(tel.keys())
'guido' in tel
```

Control flow

- if statements
- for and while loops
- range
- break and continue
- pass
- match, guards, enums

If statement

```
x = 10
y = 20
if x < y - input("threshold: "):
    print("x is less than y")
elif x == y:
    print("x equals y")
else:
    print("x is greater than y")
```

For loop

```
1  words = ['cat', 'window', 'defenestrate']
2  for w in words:
3      print(w, len(w))
```

Range

```
for i in range(5):
    print(i)

for i in range(5, 10, 3):
    print(i)
```

While loop

```
x = ['honey', 'milk', 'eggs', 'bread']
i = 0
while i < len(x):
    print(x[i])
    i += 1
```

Break and continue

```
for n in range(2, 10):
    if n % 2 == 0:
        print("Found an even number", n)
        continue
    print("Found a number", n)

for num in range(2, 10):
    if num == 5:
        break
    print("Found a number", num)
```

Pass

```
while True:
    pass  # Busy-wait for keyboard interrupt (Ctrl+C)

class MyEmptyClass:
    pass

def initlog(*args):
    pass   # Remember to implement this!
```

Match statement

```
def http_error(status):
    match status:
        case 400:
            return "Bad request"
        case 404:
            return "Not found"
        case 418:
            return "I'm a teapot"
        case 401 | 403:
            return "Not allowed"
        case _:
            return "Something's wrong with the internet"
```

Functions

- definition
- docstring
- as objects
- scoping
- input parameters
- default arguments
- keyword arguments
- special parameters

Functions

```
def fib(n):
    """Print a Fibonacci series up to n."""
    a, b = 0, 1
    while a < n:
        print(a + " ")
        a, b = b, a + b
    print()

fib(2000)
```

Functions as objects

```
1  f = fib
2  f(100)
3
4  print(fib(0))
```

Scoping

```
def scope_test():
    def do_local():
        spam = "local spam"

    def do_nonlocal():
        nonlocal spam
        spam = "nonlocal spam"

    def do_global():
        global spam
        spam = "global spam"

    spam = "test spam"
    do_local()
    print("After local assignment:", spam)
    do_nonlocal()
    print("After nonlocal assignment:", spam)
    do_global()
    print("After global assignment:", spam)
```

input parameters

```
1  def greet(name, msg):
2      print("Hello", name + ', ' + msg)
```

Default arguments

```
def ask_ok(prompt, retries=4, reminder='Please try again!'):
    while True:
        ok = input(prompt)
        if ok in ('y', 'ye', 'yes'):
            return True
        if ok in ('n', 'no', 'nop', 'nope'):
            return False
        retries = retries - 1
        if retries < 0:
            raise ValueError('invalid user response')
        print(reminder)

ask_ok('Do you really want to quit?')
```

keyword arguments

```
def parrot(voltage, state='a stiff', action='voom', type='Norwegian Blue'):
    print("-- This parrot wouldn't", action, end=' ')
    print("if you put", voltage, "volts through it.")
    print("-- Lovely plumage, the", type)
    print("-- It's", state, "!")

    parrot(1000)
    parrot(voltage=1000, action='VOOM')
    parrot(action='VOOM', voltage=1000)
    parrot('a million', 'bereft of life', 'jump')
    parrot('a thousand', state='pushing up the daisies')
```

special parameters

```
def breadshop(kind, *arguments, **keywords):
    print("-- Do you have any", kind, "?")
    print("-- I'm sorry, we're all out of", kind)
    for arg in arguments:
        print(arg)
    print("-" * 40)
    for kw in keywords:
        print(kw, ":", keywords[kw])

breadshop("White bread", "Honda Civic",
          "Toyota Corolla",
          shopkeeper="Michael Palin",
          role="Jungle",
          sketch="Bread Shop Sketch")
```

Exercise

```
def palidrome(word):
    """
    Check if a word is a palindrome.
    A palindrome is a word that reads
        the same forwards and backwards.

    Input: word (String)
    Output: Dictionary
    - length: Length of the word (int)
    - vowels: Number of vowels (int)
    - forward: The word as is (String)
    - backward: reversed (String)
    - is_palidrome:
        True if the word is a palindrome,
        False otherwise (Boolean)
    """
```

```
    """
    Example Output:
    palidrome("racecar")
    {
        length: 7,
        vowels: 3,
        forward: racecar,
        backward: racecar,
        is_palidrome: True,
    }
    """
```

This is available in codechum

Please enroll with the code

```
brazen-22018 M/Th
or
prefab-22037 T/F
```