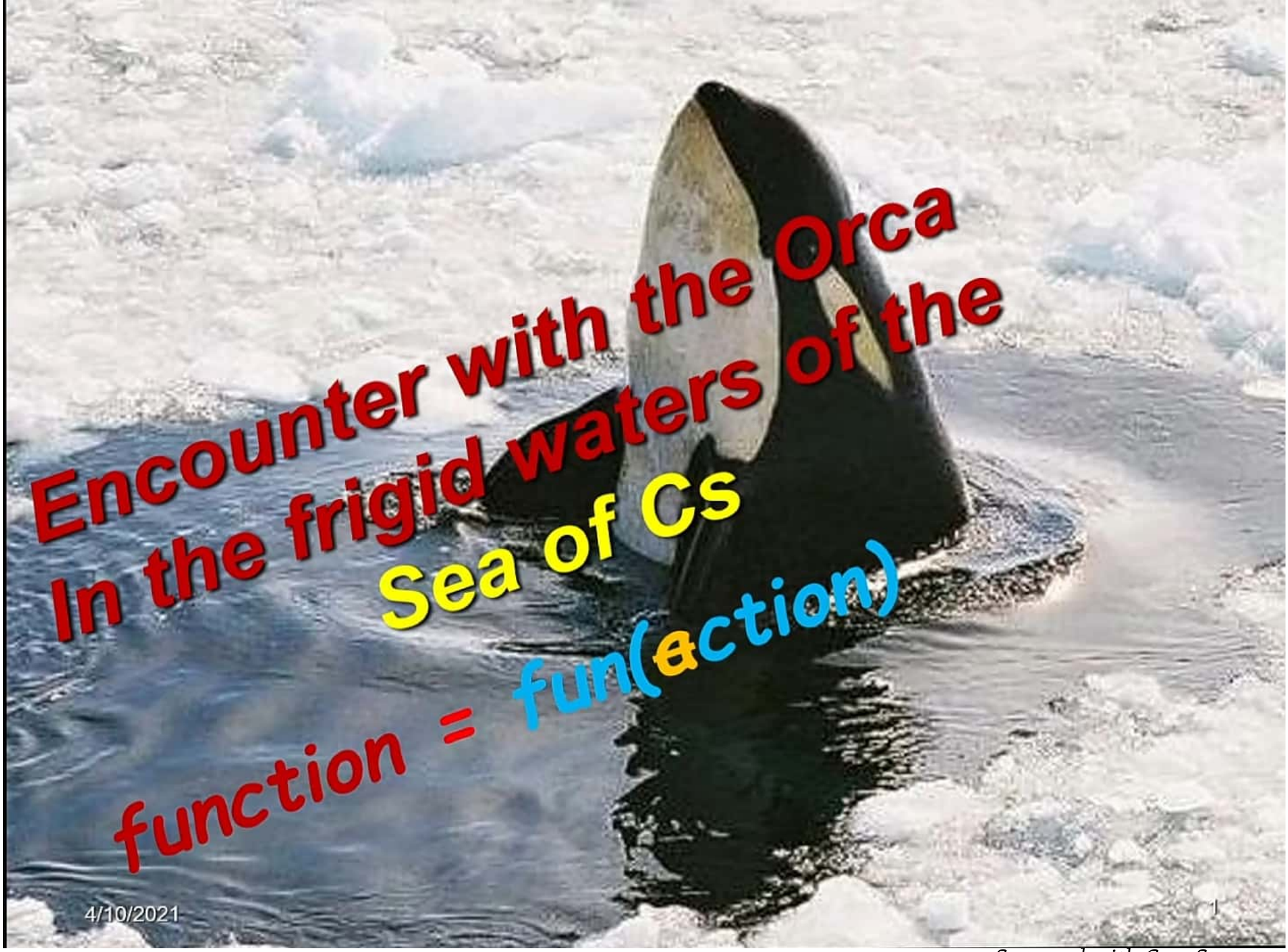

Functions

- A function is a ***self-contained block of statements*** that performs a coherent task of some kind.
- Functions ***break large computations into smaller ones***.
- Some common tasks written in the form of ***functions can be reused***.

A simple function

```
#include<stdio.h>
float max_number(float x, float y);

main( )
{
   float a, b, m;
   scanf("%f %f", &a, &b);
   m =  max_number(a,b);
   printf("Max : %f\n", m);
}

float max_number(float x, float y)
{
   if(x > y)
      return x;
   else
      return y;
}
```

Function declaration or prototype

NB: Done outside of main( )

Function call

NB: main() function ends here.
The max() function is written separately.

Function definition

One more example

```
#include<stdio.h>
int square( int  y);   /* function prototype */
int main( )
{
   int x;
   for ( x = 1; x <= 10; x++ )
      printf( "%d  ", square( x ) );
   printf( "\n" );
   return 0;
}
int square( int y )
{
   return  y * y;
}
```

main() is also a function. It returns an int

If unspecified, then by default a function returns int

What we should learn…

1. What is the control flow when a function is called?
2. Function declaration or prototype
3. Function call
4. Function definition

1. Control Flow

Execution starts with ***main()*** and when a function call like ***z = func(exp1 ,exp2, …);*** is encountered within main() then:

i. exp1, exp2, … are evaluated
ii. The values of exp1, exp2, … are passed to func( )
iii. main() is suspended temporarily
iv. func( ) is started with the supplied values
v. func( ) does the processing
vi. func( ) returns a value which is then assigned to z
vii. main( ) is resumed.

2. Function Prototype

- Every function should have a declaration which specifies the function name, its arguments types and its return type.
  - A **function declaration (prototype)** tells the compiler about the type of the arguments, their order and return type.
  - This can allow the compiler to detect errors in function call and function definition.

e.g.: `float tweet( int j, float k );`

Optionally ***j, k*** *could be omitted, that is,*

`float tweet(int, float);`

is also valid.

- This says that *tweet* is a function which takes two arguments (first one is *int* and second one is *float*) and returns a *float*.

void

- What if a function does not return any value at all?
- In this case the function returns a ***void*** and hence the return type should be ***void***.

  e.g: ***void f1(void);***
- The function ***f1()*** does not take any arguments and does not return any value too!!
- Why do we need such functions?

void

```
void nothing(void);
main( )
{
    nothing( );
}

void nothing(void)
{
    printf("When you need to do some fixed things\n
            then you can use a function like this.\n");
    return;
}
```

3. Function Definition

```
Return-type  function-name( parameters )
{
    declarations
    statements
}
```

To find maximum of three integers

```
#include <stdio.h>
int maximum( int, int, int );    /* function prototype */
int main()
{
   int a, b, c;
   printf( "Enter three integers: " );
   scanf( "%d %d %d", &a, &b, &c );
   printf( "Maximum is: %d\n", maximum( a, b, c ) );
   return 0;
}
int maximum( int x, int y, int z )
{
   int max = x;
   if ( y > max )  max = y;
   if ( z > max )  max = z;
   return max;
}
```

If you use *max* in main() it will be another variable

Declaration. When this function returns max disappears !!

NB: max is called a local variable for this function. It is not visible or usable in main()